Estimating the Resources for Quantum Computation with the QuRE Toolbox

Martin Suchara
Arvin Faruque
Ching-Yi Lai
Gerardo Paz
Frederic Chong
John D. Kubiatowicz

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2013-119
http://www.eecs.berkeley.edu/Pubs/TechRpts/2013/EECS-2013-119.html

May 31, 2013

Report Documentation Page

Report Documentation Page	*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE **31 MAY 2013**	2. REPORT TYPE	3. DATES COVERED **00-00-2013 to 00-00-2013**
4. TITLE AND SUBTITLE **Estimating the Resources for Quantum Computation with the QuRE Toolbox**		5a. CONTRACT NUMBER 5b. GRANT NUMBER 5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)		5d. PROJECT NUMBER 5e. TASK NUMBER 5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **University of California at Berkeley,Electrical Engineering and Computer Sciences,Berkeley,CA,94720**		8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSOR/MONITOR’S ACRONYM(S) 11. SPONSOR/MONITOR’S REPORT NUMBER(S)
12. DISTRIBUTION/AVAILABILITY STATEMENT **Approved for public release; distribution unlimited**		
13. SUPPLEMENTARY NOTES		
14. ABSTRACT		
15. SUBJECT TERMS		

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**116**	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

ESTIMATING THE RESOURCES FOR QUANTUM COMPUTATION WITH THE QuRE TOOLBOX

MARTIN SUCHARA[1,a], ARVIN FARUQUE[2], CHING-YI LAI[3],
GERARDO PAZ[3], FREDERIC T. CHONG[2], JOHN KUBIATOWICZ[1]

[1] *Computer Science Division, UC Berkeley, 387 Soda Hall*
Berkeley, CA 94720, U.S.A.

[2] *Computer Science Department, UC Santa Barbara, Harold Frank Hall*
Santa Barbara, CA 93106, U.S.A.

[3] *Department of Electrical Engineering Systems, University of Southern California*
Los Angeles, CA 90089, U.S.A.

This report describes the methodology employed by the Quantum Resource Estimator (QuRE) toolbox to quantify the resources needed to run quantum algorithms on quantum computers with realistic properties. The QuRE toolbox estimates a number of quantities including the number of physical qubits required to run a specified quantum algorithm, the execution time on each of the specified physical technologies, the probability of success of the computation, as well as physical gate counts with a breakdown by gate type. Estimates are performed for error-correcting codes representing codes from both the concatenated and topological code families. Our work, which provides these resource estimates for a cross product of seven quantum algorithms, six physical machine descriptions, several quantum control protocols, and four error-correcting codes, represents the most comprehensive resource estimation effort in the field of quantum computation to date.

1 Introduction

Estimating the running time, number of qubits and other resources needed by realistic models of quantum computers is the first necessary step to reducing these resource requirements. This report describes our Quantum Resource Estimator (QuRE) toolbox which we used to calculate resource estimates for a cross product of several quantum algorithms, quantum technologies, and error-correction techniques. The focus of this work is on the estimation methodology, overhead caused by error correction, and the software tools that we developed. Our toolbox simulates error correction with the Steane code [1, 2], Bacon-Shor code, Knill's post-selection scheme, and surface code, representing codes from both the concatenated and topological error-correcting code families.

The QuRE toolbox is implemented as a suite of Octave scripts. The inputs for the QuRE toolbox are the description of the physical properties of the quantum computer (such as gate error rate and gate time), and logical resource requirements of the quantum algorithms (such as number of logical qubits). The tool automatically generates resource estimates for a

[a] The corresponding author's email address is suchara@berkeley.edu.

cross product of algorithms, quantum technologies, and error-correction techniques. For each combination, the toolbox reports detailed information including the level of concatenation or code distance needed to achieve at least 50% circuit reliability, the actual circuit reliability, running time of the algorithm, number of physical qubits, and a gate count with a breakdown by gate type. For error-correcting codes that use ancilla factories, we also report the size of the ancilla factory, number of gates used by the ancilla factory, and the time needed to prepare one ancillary state.

In this work we consider a wide range of physical quantum technologies with realistic properties, each with several choices of quantum control protocols. The properties of these technologies have been studied by Hocker et al [3]. As shown in Section 3, the gate error rates for the worst physical gate in these physical technologies range from approximately 10% for Photonics I with primitive control down to 3.19×10^{-9} for ion traps with primitive control. Since the error-correction threshold for concatenated codes is typically in the 1×10^{-3} to 1×10^{-5} range, many of the models do not meet the threshold, making these concatenated error-correcting protocols unusable. On the other hand, the surface code which has threshold around 1% meets the requirements of a majority of the models.

The QuRE toolbox is preloaded with information about a variety of quantum algorithms, including binary welded tree algorithm [4], boolean formula algorithm [5], ground state estimation algorithm [6], quantum linear systems algorithm [7], shortest vector algorithm [8], quantum class number algorithm [9], and the triangle finding problem [10]. We chose these algorithms because their logical resource requirements are known and have been analyzed in the scope of IARPA's Quantum Computer Science program. The above referenced reports show the total circuit gate count, detailed breakdown by gate type, and information about parallelization for each of these algorithms. Moreover, the algorithms cover a range of quantum computation primitives such as quantum Fourier transform, quantum simulation, amplitude amplification, phase estimation, quantum walk and sieving. It should be noted that the logical resource requirements of the selected algorithms vary widely. For example, the binary welded tree algorithm requires $1,220$ logical qubits and 5.57×10^{10} logical gates, whereas the shortest vector algorithm, which is believed to be a hard problem for quantum computation, requires 4×10^{18} qubits and 2.03×10^{22} gates.

To accurately estimate the total physical resources, the QuRE toolbox heeds the locality constraints of quantum technologies – two-qubit CNOT gates can only be performed locally on two neighboring physical qubits. To that end, we use a tiled qubit layout for concatenated codes. Each tile contains physical qubits that represent the state of a single fault-tolerant logical qubit. To perform CNOT gates inside each tile, either SWAP gates or ballistic movement must be used to move the two interacting qubits together. Since movement reduction is clearly desirable, and different error-correcting codes have different structure, we use a *custom qubit layout for each of the three concatenated error-correcting codes.* For Steane code, we use the optimized qubit layout introduced by Svore et al. [11], and for the Bacon-Shor code the optimized layout of Spedalieri et al. [12]. Since there is no known optimal layout that reduces movement for the Knill's post-selection error-correcting scheme, we designed our own. QuRE also uses a tiled qubit layout for the surface code, where a pair of holes representing a logical qubit resides inside a tile. However, the computation and error correction with the surface code is inherently local, and no swapping of qubits is needed.

The ion trap technology supports reliable ballistic movement of qubits. Our resource estimation exploits the favorable properties of ballistic movement. The use of reliable ballistic movement has three benefits. First, avoiding the need to use $SWAP$ gates reduces the gate count. Second, the reliability of the move operation improves the error-correction threshold for concatenated codes, allowing to use fewer code concatenations. Finally, the move operation is much faster than a $SWAP$ gate, reducing the execution time.

This report is organized as follows. Section 2 provides a high level overview of the QuRE toolbox. In Section 3 we describe the properties of the physical technologies used by the toolbox, and in Section 4 we summarize the logical resource requirements of the studied quantum algorithms. In Section 5 we discuss the generic aspects of resource estimation that pertains to concatenated codes. We discuss tiled qubit layout, finding the optimal concatenation level, overhead associated with correcting memory errors, specifics of ballistic movement. In Section 6 we describe the specific qubit layout required by the Steane code and analyze the overhead imposed by error correction using the Steane code, and quantify the number of elementary operations required to carry out certain operations at a specified concatenation level. In Sections 7 and 8 we repeat the analysis for the Bacon Shor code and Knill's post-selection scheme. The resource requirements of quantum computation with the surface code are discussed in Sections 9 through 13. In Section 14 we provide some details about the Octave scripts that were written to implement the resource estimation. Finally, in Section 15 we show select numerical results that illustrate the resource estimation methodology developed in this document.

2 Functionality of the QuRE Toolbox

Here we present a brief overview of the functionality of the QuRE toolbox. QuRE is implemented in Octave and uses modular design to allow easy extendability.

Figure 1 shows a schematic view of the major components of QuRE. At the heart of the tool is the **Main Loop**, which iterates over all specified quantum algorithms, quantum technologies, and quantum error-correcting codes. For each combination, the Main Loop calls appropriate modules that load information about the algorithm, technology and error-correcting code.

An **Algorithm Specification** module provides information about the number of logical qubits the particular algorithm needs. *Logical* qubits are defined as the fault-tolerant qubits built of a greater number of unreliable physical qubits. Number of logical gates and simplified information about the circuit parallelism are also specified by the module. Section 4 describes how these specifications were obtained for the algorithms preloaded in the QuRE toolbox.

A **Technology Specification** module describes properties of a particular physical quantum technology. It specifies the time needed to carry out each physical gate, the error of the worst gate, and information about memory error rate per unit time. Note that by *physical* gate we shall understand a non-fault-tolerant quantum gate that is provided by the technology by executing one or more instructions. More details about the quantum technologies preloaded into QuRE that are used in this paper are in Section 3.

An **Error Correction Specification** module is provided for each supported error-correcting code. It quantifies the time and the number of physical gates needed to implement a logical gate of each type at an arbitrary level of concatenation (or, in case of topological codes, for

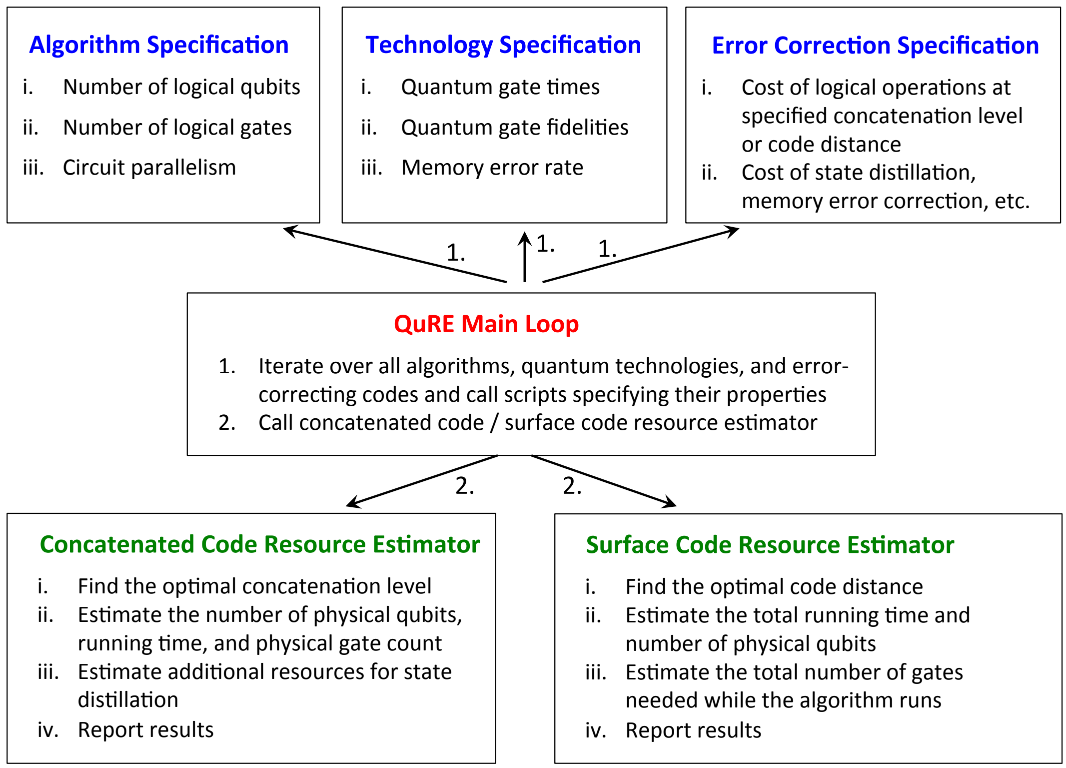


Fig. 1. Functionality of the QuRE toolbox.

an arbitrary code distance). The module also quantifies the overhead caused by magic state distillation, ancilla preparation, or any other operations pertinent to the particular error-correcting code. The methodology used to quantify these metrics for the concatenated and surface codes is described in Section 6 through 13.

The **Concatenated Code Resource Estimator** reports the resources needed by an algorithm, technology, and concatenated code loaded by the Main Loop. First, the module determines the minimum concatenation level that is sufficient to complete the algorithm successfully with high probability. Then it evaluates the resources needed to carry out fault-tolerant operations at that concatenation level, multiplies them by the number of operations used by the algorithm, evaluates additional resources needed for magic state distillation, and reports the results.

The **Surface Code Resource Estimator** reports the resources needed by the surface code and works analogously to the Concatenated Code Resource Estimator.

3 Physical Quantum Computation Architectures

The properties of the quantum technologies used in the QuRE toolbox were studied by Hocker et al. [3]. Their work describes six choices of a technology, and for some technologies they study several possible choices of a quantum control protocol. They quantify the durations and errors of all basic gates, as well as memory errors that disrupt the state of idle qubits due to interactions with the environment. The quantum technologies were modeled by Hocker et al. [3] by mapping the complicated system dynamics of each architecture onto a simplified, spin-based Hamiltonian. Noise effects were simulated with a Markovian master equation to capture dissipative and dephasing effects, while stochastic errors were used to capture control errors and environmental errors typically modeled with an open quantum system that is dynamically coupled to a bath.

Here we briefly summarize the six quantum technologies, focusing on the quantities needed for our resource estimates – the time to perform one- and two-qubit gates, and gate and memory errors. Some of the chosen models are among the most promising candidates suitable for building a large-scale quantum computer. At the same time, these models represent a range of properties that the future quantum computer may possess – for example, very fast but error prone superconductors, slower but more reliable ion traps, and neutral atoms with only average speed and average error properties due to atomic movement of qubits.

Neutral Atoms [13]: In this technology, qubits are represented by ultracold atoms trapped in an optical lattice. Light waves are used to trap and control the particles. Compared to technologies with ion traps that trap charged ions in a magnetic or optical field, the "ultracold" atom properties lead to their stability and great noise resilience. Ultracold atoms are thermally isolated and experience very slow T1 and T2 times. The only dominant errors arise from difficulties in precisely tuning the laser to a specific physical qubit position in the lattice due to atom motion inside the lattice, i.e., it is difficult to hit the moving qubit with a laser. The position offset causes an effective control error.

Superconductors [14, 15]: There are many different types of superconducting qubits. The type considered here is a superconducting phase qubit. The primitive building block for qubits is the Josephson Junction. Single- and two-qubit gates are based upon low-frequency flux pulses or shaped GHz frequency microwave pulses, and state readout is based upon a singleshot switched measurement. Superconducting qubits have very short gate times of tens of nanoseconds. State preparation is based upon reset by dissipation, and is therefore a relatively slower operation. Hocker et al. [3] attribute the relatively higher error rates to Markovian noise (a combination of radiation leakage into the Josephson junction, circuit defects, and engineering limitations upon insulating the system). The Markovian noise cannot be reduced using pulse shaping techniques that maintain constant control resources. Reducing gate times and the expense of increasing control resources can reduce such Markovian errors, but Hocker et al. considered a constrained set of control resources common to such physical system.

Ion Traps [16]: The ion trap quantum computer is based on a 2D lattice of confined ions, each of which is a physical qubit which can be moved within the lattice to accommodate local interactions between any two qubits. The ions are confined using electromagnetic field. Lasers are applied to induce coupling between qubit states to implement quantum gates. The noise terms in these laser mediated gates are associated with the intensity fluctuations of the laser, resulting in very low gate errors. While ion traps are also quite stable to environmental noise, being a charged system lends them susceptible to certain noise sources beyond those experienced by neutral atoms.

Photonics I [17,18]: This type of photonics system is essentially a form of "linear optics"

Table 1. The gate times (in ns) for all basic gate constructs.

Technology	**Control**	**CNOT**	**SWAP**	**H**	$\|+\rangle$ **prep.**	$\|0\rangle$ **prep.**	**X meas.**	**Z meas.**
Quantum Dots	Primitive	27	81	12	113	101	100	112
Neutral Atoms	Optimal	2,533	7,599	781	1,781	1,000	80,457	80,000
Neutral Atoms	Primitive	2,533	7,599	781	1,781	1,000	80,457	80,000
Neutral Atoms	Solovay Kitaev	7,710	23,130	2,075	3,075	1,000	82,075	80,000
Neutral Atoms	Trotter	11,370	34,120	2,991	3,991	1,000	82,991	80,000
Photonics I	Primitive	10	10	1	4	3	2	1
Photonics II	Primitive	12	36	1	11	10	51	50
Superconductors	Optimal	26	13	16	100	116	10	26
Superconductors	Primitive	22	17	6	100	106	16	10
Ion Traps	Dyn. Cor. Gates	1,250,000	10,000	62,000	72,000	10,000	162,000	100,000
Ion Traps	Optimal	120,000	10,000	6,000	16,000	10,000	106,000	100,000
Ion Traps	Primitive	120,000	10,000	6,000	16,000	10,000	106,000	100,000

Table 1. Continued.

Technology	**Control**	**X**	**Y**	**Z**	**S**	**T**
Quantum Dots	Primitive	10	11	1	1	1
Neutral Atoms	Optimal	457	457	915	915	915
Neutral Atoms	Primitive	457	457	915	915	915
Neutral Atoms	Solovay Kitaev	1,752	1,752	3,504	2,209	2,209
Neutral Atoms	Trotter	2,667	2,667	5,532	3,125	3,125
Photonics I	Primitive	1	1	1	1	1
Photonics II	Primitive	1	1	1	1	1
Superconductors	Optimal	10	10	1	1	1
Superconductors	Primitive	10	10	1	1	1
Ion Traps	Dyn. Cor. Gates	30,000	30,000	22,000	30,000	28,000
Ion Traps	Optimal	5,000	5,000	3,000	2,000	1,000
Ion Traps	Primitive	5,000	5,000	3,000	2,000	1,000

Table 2. The probability of error of the worst gate and the probability of an error occurring on an idle qubit for all studied quantum architectures.

Technology	**Control**	**Probability of Gate Error**	**Memory Error (per ns)**
Quantum Dots	Primitive	9.89×10^{-1}	3.47×10^{-2}
Neutral Atoms	Optimal	8.17×10^{-3}	0.00
Neutral Atoms	Primitive	8.12×10^{-3}	0.00
Neutral Atoms	Solovay Kitaev	1.77×10^{-3}	0.00
Neutral Atoms	Trotter	1.47×10^{-3}	0.00
Photonics I	Primitive	1.01×10^{-1}	9.80×10^{-4}
Photonics II	Primitive	5.20×10^{-3}	9.80×10^{-5}
Superconductors	Optimal	6.56×10^{-4}	1.00×10^{-5}
Superconductors	Primitive	1.00×10^{-5}	1.00×10^{-5}
Ion Traps	Dyn. Cor. Gates	7.99×10^{-3}	2.52×10^{-12}
Ion Traps	Optimal	2.93×10^{-7}	2.52×10^{-12}
Ion Traps	Primitive	3.19×10^{-9}	2.52×10^{-12}

quantum computing, where conventional optical equipment is used with a single element that introduces a nonlinear gate in order to generate two-qubit gates, in this case a controlled phase gate [18]. The elementary single-qubit gates are implemented in the polarization basis with wave plates. Electro-optic modulators (EOM) can apply a desired wave plate action commanded by classcial signal, i.e., voltages. Measurement is based on photon detection, which can be obtained using a single-photon counter such as avalanche photodiode (APD). A thermally excited electron can create a dark count, leading to a relatively high probability of error (approximately 10%). Hocker et al. [3] use several ancillary photons and repeated detection to achieve a sufficiently low measurement error that matches the error of other gates.

Photonics II [19]: An alternative technology to use photons as qubits abandons the linear optics paradigm and attempts to use minimal optical equipment to achieve gate operations. This technology performs two-bit gates deterministically by using the weak cross-Kerr coupling native to the optical equipment and homodyne detection [19]. This is in contrast to the photonics I approach that instead goes to great lengths to harness stronger nonlinear effects.

Quantum Dots [20]: Two electrons confined to a double well quantum dot of GaAs comprise single qubits. The logical basis are the lowest lying hyperfine energy levels of the two-qubit system, which can be tuned to a spin triple or spin singlet regime by means of an external voltage gate [20]. Controls are implemented through nearby voltage gates on each well. There have been effective implementations of these systems with primitive control [20] and dynamical decoupling. The reported errors are likely artificially too high because of approximations used in simulating two-qubit gates, and difficulty with accurately simulating the noise models. The reported errors are too high to make it usable with any error-correcting scheme in our work. The studies of this technology are still ongoing, and results will likely improve in further works.

In the models discussed above, the basic quantum gate set includes the following operations: controlled not (CNOT), swapping of two qubits (SWAP), the Hadamard gate (H), state preparation ($|+\rangle$ prep., $|0\rangle$ prep.), qubit measurement (X meas., Z meas.) the Paulis (X, Y, Z) and the S and T gates (S, T). We use this gate set throughout this document. Note that the gate set is universal for quantum computation, and is overinclusive. All the quantum technologies either support these operations natively, or the gates can be constructed from more elementary operations. For example, SWAP can be constructed using three CNOT gates. Note that there is often more than one way to compose a gate. For example, an alternative way of decomposing the $SWAP$ gate is to use an $iSWAP$, $CPHASE$, and two (parallel) S gates. This particular decomposition can yield substantially better error rate and gate time – for example, this takes a total of 17 ns on the superconducting architecture, compared to the 66 ns needed to perform three $CNOT$s, reducing both the gate time and error. The report of Hocker et al. describes the optimal choices for these decompositions.

The durations of basic one- and two-qubit gates are shown in Table 1. Note that all units of time in this paper are in nanoseconds, unless stated otherwise. Another important property of the technologies is reliability. Table 2 summarizes the error probability after applying the worst gate, as well as the probability of a bit flip per nanosecond on an idle qubit.

The models of Hocker et al. [3] consider many details that a realisitic computer must posses, including a basic instruction set, errors due to qubit movement and decoherence, and the use of currently known control protocols to optimize properties of quantum gates. While the experimental demonstrations of these technologies to date have been limited to small scale, the models used here are among the most plausible and realistic for a large scale quantum computer.

4 Quantum Algorithms

In order to study the resource requirements of quantum computation, we need to use representative quantum algorithms with the right "mix" of quantum gates and known parallelism properties. For this purpose, the following algorithms were studied in the IARPA Quantum Computer Science program, and we report their properties here:

- binary welded tree algorithm [4] which finds the opposite root of two connected binary trees,
- boolean formula algorithm [5] which evaluates a boolean formula,
- ground state estimation algorithm [6] which finds the ground state energy of a molecule,
- quantum linear systems algorithm [7] which finds x in the linear system $Ax = b$,
- shortest vector algorithm [8] which finds unique shortest vector in an integer lattice,
- quantum class number algorithm [9] which finds the order of the class group of a real quadratic field, and
- the triangle finding problem [10] which finds the nodes forming a triangle in a dense graph.

These algorithms represent several key algorithmic techniques, including e.g. quantum Fourier transform, quantum simulation, amplitude amplification, quantum random walk, quantum simulation, and phase estimation. These techniques form the building blocks of many other quantum algorithms.

The resource requirements reported in this section use problem sizes specified in the IARPA Computer Science program, and are studied in [4–10]. The problem sizes in the IARPA program were chosen so that they are untractable for classical computers. Therefore, it should not be surprising that some of the studied algorithms may be intractable for quantum computers as well. We describe the algorithms and the problem sizes next.

4.1 Description of the Quantum Algorithms

Binary Welded Tree [21]: The problem input is a pair of binary trees that are connected at their leaves. The goal is to start at a root of one of the trees marked as 'entrance' and find the opposite root marked as 'end' by traversing the graph. The algorithm uses an oracle to discover the structure of the graph. The oracle returns the names of the three neighbors of a specified vertex. Due to a careful construction of the problem (each node has an exponential number of labels, trees are connected in a specific way, etc.) a sub-exponential classical algorithm that finds the 'end' w.h.p. is not known. A quantum algorithm that uses continuous-time quantum random walk finishes in polynomial time. In order to evaluate a problem instance that is intractable for classical computers, the tree depth was chosen as $n = 300$.

Boolean Formula [22]: This problem uses the algorithm of Childs et al. [22] for solving the boolean formula problem which determines if an $\{AND, OR\}$ tree evaluates to true. The boolean formula algorithm is used to find the best overall strategy in a two-player game of hex with a 9 by 7 board. The sequence of moves in a two-player game can be represented by an $\{AND, OR\}$ tree where the leafs correspond to the outcomes of the game. The algorithm is based on discrete-time quantum walk. An oracle indicates which of the two players wins.

Ground State Estimation [23]: This algorithm finds the ground state energy, E_0, of a specified molecule. The energy is estimated to b bits of precision. The quantum circuit for the algorithm is studied in [6] which quantifies the logical gate counts in the quantum circuit as a function of the precision b and the number of wave functions M that describe the ground state. The polynomial time quantum algorithm is based on the approach outlined in [23] which uses quantum simulation and phase estimation. In this work, we chose a problem instance that finds E_0 for the Fe_2S_2 molecule with $b = 9$ bits of precision. Note that the Fe_2S_2 molecule needs $M = 208$ wave functions to describe the ground state.

Quantum Linear Systems [24]: The algorithm finds the solution to a linear system $Ax = b$ by mapping it into a quantum system $A|x\rangle = |b\rangle$ with state vectors $|x\rangle$ and $|b\rangle$ and Hamiltonian A. Quantum phase estimation and Fourier transform are used to solve for $|x\rangle$ by extracting the eigenvalues of A. The studied problem size is $dim(A) = 3 * 10^8$. The report [7] analyses a deterministic variant of the quantum linear systems algorithm where a non-deterministic measurement is replaced by estimating probabilities using amplitude estimation.

Shortest Vector Algorithm [25]: Given a $n \times n$ integer lattice B, the algorithm finds an integer vector v such that the vector Bv has minimal length under the Euclidean norm. The problem formulation also guarantees that the shortest vector is unique – the next shortest vector is longer by a factor of at least n^3. The conceptual primitives used by the quantum algorithm are quantum Fourier transform and sieving. This algorithm is not efficient because it runs in a time longer than poly(n). Nevertheless, it is interesting because it uses a variety of primitives, some of which are not explored by the other quantum algorithms. The instance size was chosen as $n = 50$.

Quantum Class Number [26]: This algorithm finds the order of the class group of a real quadratic number field. The analysis is based on the work of Hallgren [26] that shows how the class number of a real quadratic number field may be computed by extending a standard algorithm for the hidden subgroup problem. The algorithm also uses quantum Fourier transform. The size of the problem was chosen to be $n = 124$ decimal digits in the quadratic discriminant.

The Triangle Finding Problem [27]: Given a dense graph, this algorithm finds the nodes in the graph forming a triangle if one exists. Similarly as in the case of the binary welded tree algorithm, an oracle and a specific graph structure is used to ensure the problem does not have an efficient classical solution. The graph is dense, containing fully connected sub-components, and only one triangle, if any. The conceptual primitives used by the efficient quantum algorithm are quantum random walk and amplitude amplification. In this work we chose a graph with $32,768$ nodes.

4.2 Methodology for Logical Resource Estimation

The algorithm analyses in [4–10] report logical gate counts that contain discrete quantum gates as well as arbitrary rotations. Each of these arbitrary rotations needs to be decomposed into a discrete set of gates that can be implemented fault tolerantly. To perform this decomposition, we use the result of Bocharov et al. [28] that shows how to obtain minimal depth decomposition in the $\{H, T\}$ basis, ensuring that the occurrence of expensive T gates is minimized. Their approach is based on the Solovay-Kitaev theorem [29].

The result of Bocharov et al. [28] states that for any ϵ, an arbitrary single-qubit gate U can be decomposed with precision ϵ using $\Theta(log^c(1/\epsilon))$ gates from the universal discrete gate set. Figure 3 in [28] shows that a decomposition that uses $1,000$ T gates results in gate error of 1×10^{-7}. Furthermore, increase of the number of T gates in the decomposition by one order of magnitude improves the error by three orders of magnitude. Our QuRE toolbox uses the results of this empirical study. We first determine the desired precision of the decomposition so that 50% probability of success of the algorithm can be guaranteed. Then we quantify the number of H and T gates per rotation.

Let *arbitraryRot* denote the number of arbitrary rotations in the algorithm, and let *errorPerRot* be the desired maximal error for each arbitrary rotation. To guarantee that the accumulated error across all rotations is at most 0.5, it is sufficient to require:

$$errorPerRot \leq 0.5/arbitraryRot \tag{1}$$

Then from Figure 3 in [28] the number of H and T gates per rotation is approximately:

$$gatesPerRot = 10^{\frac{2-log_{10}(errorPerRot)}{3}}. \tag{2}$$

4.3 The Logical Resource Requirements

The summary of the properties of the quantum algorithms appears in Tables 3, 4, and 5. Note that the data in these tables summarizes the *logical* resource requirements of the algorithms

Table 3. Logical gate count for each algorithm.

Algorithm	**CNOT**	**H**	$\|+\rangle$ **prep.**	$\|0\rangle$ **prep.**	**Z meas.**	**X**	**Z**	**S**	**T**	**Rotation**
Binary Welded Tree	2.54×10^{10}	4.65×10^{9}	6.00×10^{8}	0	302	4.20×10^{8}	0	0	2.46×10^{10}	1.01×10^{9}
Boolean Formula	1.30×10^{27}	8.93×10^{24}	0	3.48×10^{24}	6	2.17×10^{24}	0	1.61×10^{19}	3.12×10^{25}	3.54×10^{3}
Class Number	1.15×10^{18}	3.85×10^{17}	0	0	0	1.88×10^{17}	0	0	1.35×10^{18}	0
Ground State Est.	2.18×10^{16}	1.51×10^{15}	0	220	12	0	5.04×10^{14}	5.04×10^{14}	0	2.56×10^{14}
Quant. Linear Syst.	1.30×10^{25}	1.70×10^{23}	0	1.10×10^{24}	56	2.30×10^{5}	2.10×10^{5}	1.70×10^{23}	2.80×10^{23}	2.80×10^{25}
Shortest Vector	1.00×10^{22}	3.00×10^{20}	0	1.70×10^{18}	7.00×10^{16}	2.00×10^{18}	0	0	1.00×10^{22}	1.00×10^{17}
Triangle Finding	8.00×10^{13}	2.20×10^{13}	0	0	1.40×10^{5}	7.30×10^{12}	0	1.10×10^{13}	7.70×10^{13}	0

Table 4. Parallelization factor for each gate type.

Algorithm	**CNOT**	**H**	$\|+\rangle$ **prep.**	$\|0\rangle$ **prep.**	**Z meas.**	**X**	**Z**	**S**	**T**	**Rotation**
Binary Welded Tree	302	53	1.20×10^{3}	1	302	302	1	1	1	302
Boolean Formula	1.22	1.22	1	1.13	60	1.81	1	1	1.91	1
Class Number	1	1	1	1	1	1	1	1	1	1
Ground State Est.	1.5	6	1	220	1	1	3	3	1	1.5
Quant. Linear Syst.	1	26	1	178	14	1	1	1	1	65
Shortest Vector	1	1.00×10^{8}	1	1.80×10^{9}	250	1	1	1	1	250
Triangle Finding	1	1	1	1	4.62×10^{4}	1.01	1	1	1	1

Table 5. The number of logical qubits required by each algorithm.

Algorithm	**Logical Qubits**
Binary Welded Tree	1.22×10^3
Boolean Formula	2.63×10^3
Class Number	1.88×10^{17}
Ground State Est.	2.20×10^2
Quant. Linear Syst.	2.55×10^2
Shortest Vector	4.00×10^{18}
Triangle Finding	9.04×10^7

before error-correction overhead is taken into account. In particular, Table 5 shows the number of logical qubits needed by each algorithm. This count includes all ancillas. Table 3 shows the number of logical quantum gates of each type needed to implement each algorithm. Finally, Table 4 shows the parallelization factor for each gate type. The parallelization factor indicates how many of the gates of a particular type can be performed in parallel. For example, in case of the ground state estimate algorithm, we see that 1.51×10^{10} logical Hadamard gates are needed, and the parallelization factor is 6, meaning that on average 6 H gates can be scheduled simultaneously in the logical quantum circuit.

5 Error Correction with Concatenated Codes

This section describes our high level approach to error correction with the concatenated codes. First in Subsection 5.1 we introduce a qubit layout that uses tiles as building blocks, and describe the functionality of a single tile. Then in Subsection 5.2 we explain why it is necessary to customize the qubit layout in each tile to meet the specific requirements of each error-correcting code / quantum technology combination. In Subsection 5.3 we explain how to determine the optimal number of concatenation levels of error correction as well as how to estimate the success probability of the algorithm. Finally, in Subsection 5.4 we quantify the resources needed to correct memory errors and in Subsection 5.5 we justify our choice of syndrome extraction method.

5.1 The Tiled Qubit Layout

Our qubit layout for concatenated codes is modeled after the microarchitecture of Svore et al. [11], and Spedalieri et al. [12]. In particular, we use a block structure, where each building block (tile) stores a logical qubit. Each tile contains enough space to store one data qubit, one ancilla, sufficient number of verification qubits (to allow ancilla verification), and space for all data qubits from one neighboring tile. Error correction can be performed in each tile by applying the correct gate sequence. Communication between tiles is achieved by swapping. For example, to perform a $CNOT$ operation on two neighboring tiles, the data qubits from one tile are moved into the other tile, the $CNOT$ operation is performed, and then the qubits return to their original location.

A high level picture of the architectural organization is in Figure 2. We assume that the tiles are arranged in a 2-D structure. An example of the structure of a tile is shown in Figure 3. The figure shows the layout of Svore et al. [11] which has tile size 6×8. At higher levels of concatenation, the structure is expanded in a hierarchical fashion using 6×8 building blocks

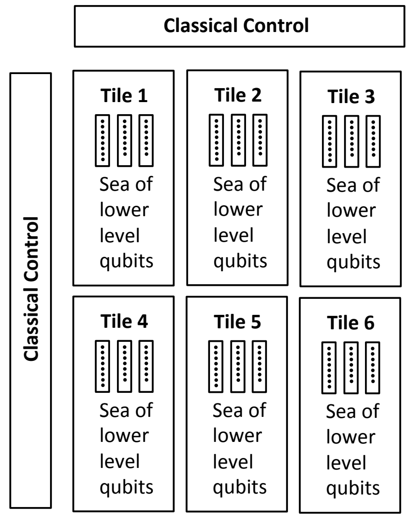


Fig. 2. The qubit layout consists of tiles. Each tile represents one logical qubit.

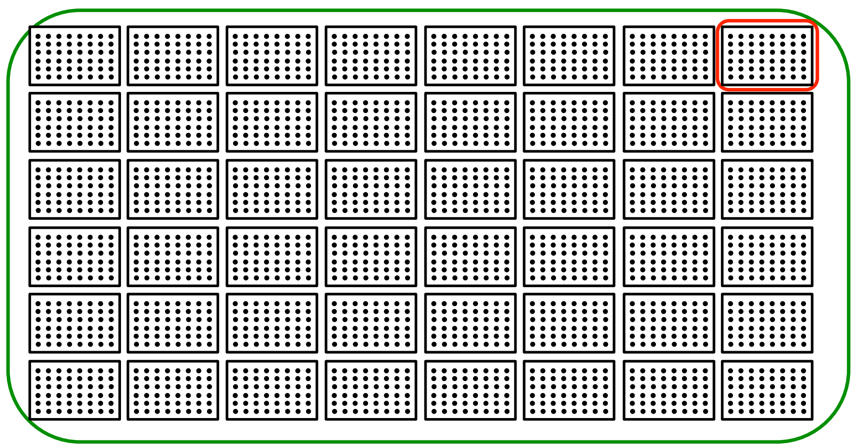


Fig. 3. One tile at the second concatenation level (bounded by the large green box). A tile at the first level of concatenation consists of the qubits in the smaller red box.

from the next lower level. Note that the tile size will differ for different concatenated codes as enough qubits must be present to store one logical data qubit as well as ancillas needed to perform error correction on the data.

5.2 Customizing Qubit Layout

The layout needs to be customized for each concatenated code. The Steane code uses a tile of size 6×8 whereas the Bacon Shor code uses a tile of size 7×7. The qubit layout and sequence of operations needed to perform error correction was optimized to maximize the error-correction threshold and minimize movement and number of operations in [11, 12]. Therefore, we use the same layout. Since the optimal layout for the Knill's post-selection scheme hasn't been studied, we had to design our own tile and sequence of operations.

5.3 Finding the Optimal Concatenation Level

Here we answer the question how many levels of concatenation we need to achieve a desired fidelity. Let's assume that p is the gate error, the failure probability of components at the lowest level of the code. Fault-tolerant constructions of the gates guarantee that the probability that a circuit introduces two errors is $O(p^2) = cp^2$ if no concatenation is used. Here $c = 1/p_{th}$ is a constant representing the threshold of the concatenated code. It follows that with two levels of concatenation, the probability of failure becomes $c(cp^2)^2$, and with k levels of concatenation $(cp)^{2^k}/c$. Suppose that we wish to simulate a circuit with N gates and achieve final accuracy of ϵ. Thus each gate must be accurate to ϵ/N so we need to find k satisfying: $\frac{(cp)^{2^k}}{c} \leq \frac{\epsilon}{N}$.

In this work we assume that the circuit needs to finish with success probability of at least 50%. Therefore, the desired concatenation level is $k = \lceil log(\frac{log(0.5/(N*p_{th}))}{log(p/p_{th})})/log(2) \rceil$, and the probability with which the circuit outputs the correct result is $p_{success} = (1 - p_{th}(\frac{p}{p_{th}})^{2^L})^N$. Note that the success probability $p_{success}$ will be generally in the 50% to 100% range due to the use of the ceiling function when calculating the desired concatenation level k.

5.4 Correcting Memory Errors

Error correction needs to be performed periodically even for idle qubits on which no quantum gates act. As was shown in Section 3, the memory error rate for a single qubit ranges up to $p_{mem} = 3.47 \times 10^{-2}$ per ns. To estimate the resources needed to correct memory errors, we assume that these errors are corrected periodically for all qubits, and calculate the optimal amount of time T between subsequent error-correction steps. This approach should not lead to significant overestimation of resources because our algorithm analysis shows that most of the qubits are idle most of the time, and successive gates that are always followed by error-correction operations frequently act on the same set of qubits.

Error correction needs to be performed early enough to ensure that the probability of an error on any single qubit is below the gate errror probability p. Thus we require $T = p/p_{mem}$.

5.5 Choice of Syndrome Extraction

The three possible error syndrome extraction methods are Knill, Shor, and Steane. We use the Steane extraction method for the Steane and Bacon-Shor codes and Knill's method for the Knill's post-selection scheme for the following reasons:

1. **Steane code:** for our qubit layout, it is important to minimize the use of space and thus movement, potentially at the cost of increased waiting and memory errors. The reason is that memory errors will typically occur at lower rates than $SWAP$ gate errors. This leaves us with the choice of the Shor's and Steane's method, because Knill's method requires two ancillas, and hence our tiles would have to be much bigger, require more operations per gate, more physical qubits, and all operations would take longer. Of the remaining two, Shor's method is slightly more space efficient if implemented fully sequentially. However, in Shor-type syndrome extraction, the number of gates between data and ancilla qubits is greater than in Steane-type syndrome extraction, which will negatively affect the threshold, requiring more concatenations, and more frequent correction of memory errors on idle qubits. Therefore, we believe that Steane's method is

superior to the other two. Steane syndrome extraction was chosen for similar reasons in [11].

2. **Bacon-Shor code:** We chose Steane syndrome extraction for the same reasons as above.

3. **Knill's post-selection scheme:** The Knill's post-selection scheme requires the use of Knill's method.

6 Error Correction with the Steane Code

Now we describe the overhead imposed by error correction using the Steane $[[7, 3, 1]]$ code [1,2]. First in Subsection 6.1 we provide some notation and explain how gates can be implemented fault tolerantly. Then in Subsection 6.2 we describe the tile size and qubit locations within a tile optimized for the Steane code. Then in Subsection 6.3 we provide resource estimates for each logical gate assuming that the Steane code uses m levels of concatenation. Note that numerical results appear in Section 15 at the end of the document. There we illustrate the resource estimation methodology developed in this section by showing the resources used by elementary logical gates at different concatenation levels, as well as the total resources needed by each algorithm on all the studied physical technologies.

6.1 The Steane Code

Logical qubits in the Steane code are encoded using seven qubits. The Steane Code is a stabilizer code [30] with stabilizers $g_1 = IIIXXXX$, $g_2 = IXXIIXX$, $g_3 = XIXIXIX$, $g_4 = IIIZZZZ$, $g_5 = IZZIIZZ$, $g_6 = ZIZIZIZ$ that map the encoded logical states to themselves. Therefore, the logical states $|\bar{0}\rangle$ and $|\bar{1}\rangle$ can be written as follows: $|\bar{0}\rangle = \frac{1}{\sqrt{8}}[|0000000\rangle + |1010101\rangle + |0110011\rangle + |1100110\rangle + |0001111\rangle + |1011010\rangle + |0111100\rangle + |1101001\rangle]$, and $|\bar{1}\rangle = \frac{1}{\sqrt{8}}[|1111111\rangle + |0101010\rangle + |1001100\rangle + |0011001\rangle + |1110000\rangle + |0100101\rangle + |1000011\rangle + |0010110\rangle]$. Note that the states $|\bar{0}\rangle$ and $|\bar{1}\rangle$ were chosen to have even and odd number of ones, respectively.

The Steane code can be concatenated, each Level m encoded qubit block is built using seven Level $m - 1$ logical qubits and gates. Level 1 blocks are at the lowest level of encoding and they are built of Level 0 qubits and gates provided at the physical level.

We will use the following notation. Standard gates at the m-th level of concatenation are denoted by $X_{(m)}$, $Y_{(m)}$, $Z_{(m)}$, $CNOT_{(m)}$, $H_{(m)}$, $S_{(m)}$, $T_{(m)}$. Measurement in the X and Z basis will be denoted by $\mathcal{M}_{X(m)}$ and $\mathcal{M}_{Z(m)}$. Fault-tolerant preparation of states $|0\rangle$ and $|+\rangle$ at the m-th level of concatenation is denoted by $P_{|0\rangle(m)}$ and $P_{|+\rangle(m)}$. The error-correction operation is represented by $\mathcal{EC}_{(m)}$ and error detection is represented by $\mathcal{ED}_{(m)}$. In our analysis, the sequence of operations required to implement a specified gate or operation is denoted $ops(...)$, and the time required to perform the specified operation when parallelization is taken into account is denoted $time(...)$ where ... is replaced by the operations we wish to analyze.

It is easy to verify that the two circuits in Figure 4 produce the logical states $|\bar{0}\rangle$ and $|\bar{+}\rangle$. However, these circuits are not fault tolerant as a single fault will lead to preparation of an incorrect state. The fault-tolerant implementation of these circuits is shown in Figure 5. To prepare state $|\bar{0}\rangle$, both the data code block $|Q\rangle$ and an ancilla $|V\rangle$ are non-fault tolerantly

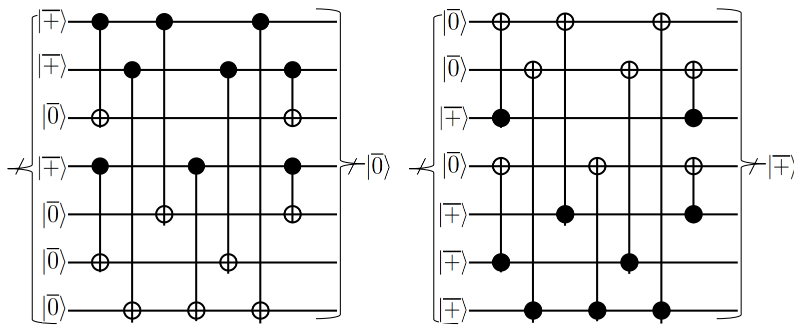


Fig. 4. Non-fault-tolerant circuit for preparation of state $|\bar{0}\rangle$ on the left and $|\bar{+}\rangle$ on the right.

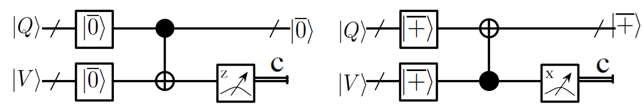


Fig. 5. Fault-tolerant preparation of state $|0\rangle$ and $|+\rangle$ must be repeated until the measurement outcome c is 1.

initialized in state $|\bar{0}\rangle$. A fault in the preparation is detected by performing a $CNOT$ on the two code blocks and measuring the ancilla in the Z basis. If the measurement outcome is -1 an error occurred and the process must be repeated. The state $|\bar{+}\rangle$ can be prepared similarly as shown on the right hand side of the figure.

The Steane code belongs to the family of Calderbank-Shor-Steane (CSS) codes, a family of codes with transversal implementation of most gates, including the $CNOT$ gate. Transversal $CNOT$ gate at Level m can be obtained by applying seven $CNOT$ gates to the corresponding control and target qubits at Level $m-1$. Figure 6 shows a fault-tolerant implementation of the $CNOT$ gate. Gates that can be performed transversally also include the single qubit Pauli gates and S and H gates. Figure 6 also shows how to do measurement. Since the logical state $|\bar{0}\rangle$ and $|\bar{1}\rangle$ have even and odd number of ones, respectively, the classical parity calculation on the Z basis measurements of the seven code blocks distinguishes $|\bar{0}\rangle$ and $|\bar{1}\rangle$. To ensure fault tolerance, each logical gate is followed by error correction.

In order to have a universal gate set, we also need the $\pi/8$ gate called the T gate. Unfortunately, a fault-tolerant version of this gate cannot be constructed transversally. A fault-tolerant T gate is shown in Figure 7. This gate sequence was originally constructed in [31] using one-bit teleportation. The gate sequence teleports the state $|\psi\rangle$ from the data block to the ancilla and applies the T gate to the state. Note that the ancilla must be fault-tolerantly initialized in the state $|\phi_+\rangle = TH\,|0\rangle = \frac{|0\rangle + e^{i\pi/4}|1\rangle}{\sqrt{2}}$. This initialization is shown in the dashed block, and requires two fault-tolerant preparations of the cat state $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$, which is depicted in Figure 8.

To correct errors, we use the Steane error extraction method [2,32] that can be better parallelized and uses fewer gates than the cat state method [32]. A fault-tolerant implementation of the Steane method is shown in Figure 9. Two ancillas are prepared fault-tolerantly, one in state $|+\rangle$ and the other in state $|0\rangle$. X errors are corrected first. The first $CNOT$ does not affect the encoded state of the ancilla $|+\rangle$ or the encoded code block, but it propagates each

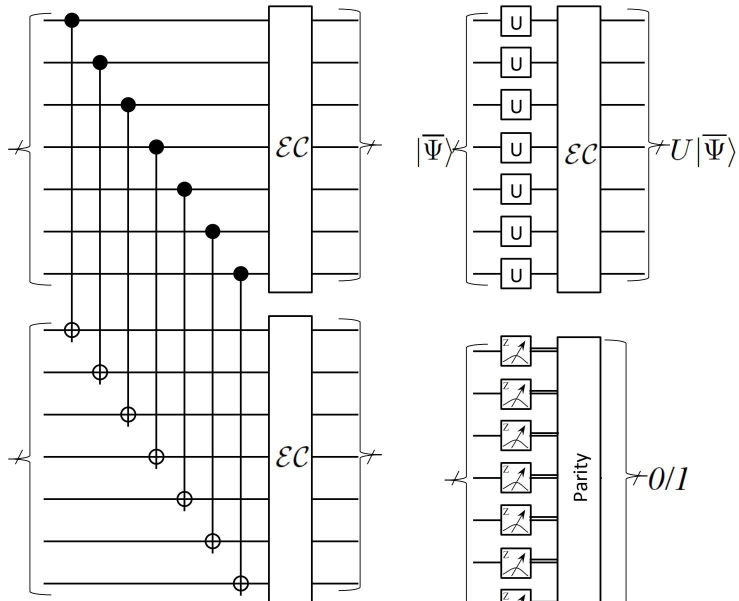


Fig. 6. Fault-tolerant implementation of the $CNOT$ gate (on the left), the single qubit gates X, Y, Z and S (top right corner), and Z basis measurement (bottom right).

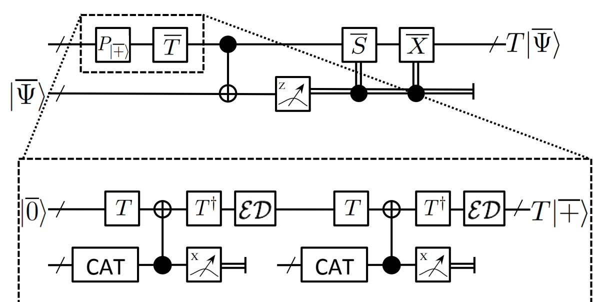


Fig. 7. Fault-tolerant T gate construction.

X error in the data to the corresponding position on the $|+\rangle$ ancilla. The Z-type syndrome which detects X errors is extracted by measuring the ancilla qubits in the Z basis and applying a classical parity check to the measurement outcomes. E.g., for a stabilizer $\bigotimes_{i=1}^{n} Z^{b_i}$ and measurement outcome $(z_1, z_2, ..., z_n)$ the eigenvalue of the stabilizer is $b \cdot z$. The X-type error correction is then performed as indicated by the syndromes, this is represented in the circuit in Figure 9 by the symbol $\mathcal{R}_X$. The X-type stabilizers are obtained similarly by coupling the data to the ancilla initialized in state $|0\rangle$, applying a $CNOT$ and X basis measurement followed by the parity check and Z-type error correction, if any.

Note that the error detection operation denoted $\mathcal{ED}$ differs from the error correction $\mathcal{EC}$ depicted in Figure 9 by removing the two error-correction operations $\mathcal{R}_X$ and $\mathcal{R}_Z$.

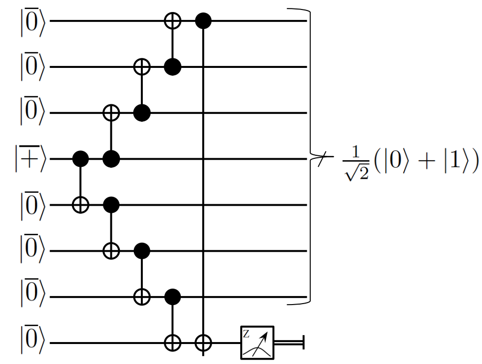


Fig. 8. Fault-tolerant circuit for cat state preparation.

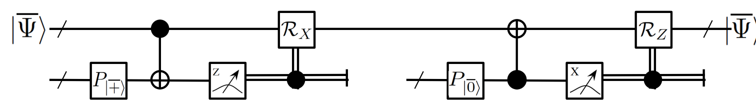


Fig. 9. Illustration of the Steane-EC syndrome extraction method.

6.2 Tiled Layout for the Steane Code

In order to appropriately quantify the resources required, we first describe the physical layout being used. We use the tile structure used in [11], designed to minimize the amount of $SWAP$ operations used during error-correction routines, and thus preserve a high error threshold. The tile consists of a 6×8 lattice of qubits. The following figure shows a snapshot of the operations and state of the tile during the ancilla preparation part of error correction:

$$\begin{array}{cccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & d_6 & 0 & d_5 & 0 & d_3 & 0 & 0 \\
0 & 0 & v_3 & a_3 & a_6 & 0 & 0 & 0 \\
0 & 0 & v_2 & a_5 & a_4 & a_1 & 0 & 0 \\
0 & 0 & v_1 & a_2 & 0 & a_7 & 0 & 0 \\
0 & d_4 & 0 & d_2 & 0 & d_1 & 0 & d_7
\end{array} \tag{3}$$

Ancilla qubits are labeled with a and data qubits are labeled with d. Ancilla qubits labeled with v are used for verification. O locations represent dummy qubits that are used as channels when qubits are swapped. For details on the exact gate sequences being used we refer the reader to the original paper [11]. The threshold obtained for this layout, and the corresponding circuits implementing the error-correction subroutines, is $p_{\text{th}} = 3.6 \times 10^{-5}$ which compares favorably to the idealized threshold $p_{\text{th}} = 1.85 \times 10^{-5}$ that ignores the cost of movement [11].

It is easy to calculate the number of physical qubits in the system. In a system with n tiles and Steane code with l levels of concatenation, the number of physical qubits we need is $n(8 \times 6)^l$. Note that the number of physical qubits is different when the Bacon code and the Knill scheme are used. First, these codes use tiles of different size. Second, these codes require the use of ancillas, and additional space is needed for ancilla factories. We provide estimates of the size of ancilla factories in our Bacon-Shor code and Knill scheme analysis in the subsequent sections.

6.3 Quantifying Resources with m Levels of Concatenation

Here we estimate the resources needed to perform a single logical gate at Level m of concatenation with the Steane code. Specifically, we calculate the number of Level 0 gates, and the total gate time taking parallelization into account. We distinguish horizontal and vertical $CNOT$ gates, horizontal and vertical $CNOT$s act on qubits (or tiles) that are adjacent along the horizontal and vertical axes, respectively. Similarly we distinguish horizontal and vertical $SWAP$ gates. We express the resources for a single logical gate at Level m:

Error Detection and Correction

A fault-tolerant implementation of qantum error-correcting protocol for a single logical code block (assuming ancilla preparation always succeeds):

$$\begin{aligned} ops(\mathcal{EC}_{(m)}) &= (4+4+6)hCNOT_{(m-1)} + (21+7)vCNOT_{(m-1)} + 7H_{(m-1)} \\ &+ 18hSWAP_{(m-1)} + 15vSWAP_{(m-1)} + 8P_{|+\rangle(m-1)} + 12P_{|0\rangle(m-1)} + 20\mathcal{M}_{X(m-1)} \end{aligned} \tag{4a}$$

$$\begin{aligned} time(\mathcal{EC}_{(m)}) &= 2max(P_{|+\rangle(m-1)}, P_{|0\rangle(m-1)}) \\ &+ 2max(P_{|+\rangle(m-1)}, P_{|0\rangle(m-1)}, hCNOT_{(m-1)}, vCNOT_{(m-1)}) \\ &+ 2max(hSWAP_{(m-1)}, vSWAP_{(m-1)}, vCNOT_{(m-1)}, P_{|0\rangle(m-1)}) \\ &+ 2max(hSWAP_{(m-1)} + hCNOT_{(m-1)}, vCNOT_{(m-1)}) \\ &+ 2max(hSWAP_{(m-1)}, hCNOT_{(m-1)}) \\ &+ 2max(hSWAP_{(m-1)}, vSWAP_{(m-1)}, hCNOT_{(m-1)}, \mathcal{M}_{X(m-1)}) \\ &+ 2max(hSWAP_{(m-1)}, vSWAP_{(m-1)}, \mathcal{M}_{X(m-1)}) \\ &+ 2vCNOT_{(m-1)} + 2\mathcal{M}_{X(m-1)} \end{aligned} \tag{4b}$$

Error detection for a single logical code block:

$$ops(\mathcal{ED}_{(m)}) = \mathcal{EC}_{(m)} \tag{5a}$$

$$time(\mathcal{ED}_{(m)}) = \mathcal{EC}_{(m)} \tag{5b}$$

Fault-Tolerant Horizontal and Vertical CNOT Gate

$$ops(hCNOT_{(m)}) = 7vCNOT_{(m-1)} + 112hSWAP_{(m-1)} + 14vSWAP_{(m-1)} + \mathcal{EC}_{(m)} \tag{6a}$$

$$ops(vCNOT_{(m)}) = 7vCNOT_{(m-1)} + 12hSWAP_{(m-1)} + 70vSWAP_{(m-1)} + \mathcal{EC}_{(m)} \tag{6b}$$

$$\begin{aligned} time(hCNOT_{(m)}) &= max(hSWAP_{(m-1)}, vSWAP_{(m-1)}) + 6hSWAP_{(m-1)} \\ &+ vCNOT_{(m-1)} + max(P_{|+\rangle(m-1)}, P_{|0\rangle(m-1)}, hSWAP_{(m-1)}) \\ &+ max(P_{|+\rangle(m-1)}, P_{|0\rangle(m-1)}, hCNOT_{(m-1)}, vCNOT_{(m-1)}, hSWAP_{(m-1)}) \\ &+ max(P_{|+\rangle(m-1)}, P_{|0\rangle(m-1)}) + max(P_{|+\rangle(m-1)}, P_{|0\rangle(m-1)}, hCNOT_{(m-1)}, vCNOT_{(m-1)}) \\ &+ 2max(hSWAP_{(m-1)}, vSWAP_{(m-1)}, vCNOT_{(m-1)}, P_{|0\rangle(m-1)}) \\ &+ 2max(hSWAP_{(m-1)} + hCNOT_{(m-1)}, vCNOT_{(m-1)}) \\ &+ 2max(hSWAP_{(m-1)}, hCNOT_{(m-1)}) \\ &+ 2max(hSWAP_{(m-1)}, vSWAP_{(m-1)}, hCNOT_{(m-1)}, \mathcal{M}_{X(m-1)}) \\ &+ 2max(hSWAP_{(m-1)}, vSWAP_{(m-1)}, \mathcal{M}_{X(m-1)}) + 2vCNOT_{(m-1)} + 2\mathcal{M}_{X(m-1)} \end{aligned} \tag{6c}$$

$$\begin{aligned}
&time(vCNOT_{(m)}) = max(vSWAP_{(m-1)}, hSWAP_{(m-1)}) \\
&+ 2max(vSWAP_{(m-1)}, vCNOT_{(m-1)}) + 4vSWAP_{(m-1)} \\
&+ max(P_{|+\rangle(m-1)}, P_{|0\rangle(m-1)}, hSWAP_{(m-1)}, vSWAP_{(m-1)}) \\
&+ max(P_{|+\rangle(m-1)}, P_{|0\rangle(m-1)}, hCNOT_{(m-1)}, vCNOT_{(m-1)}, hSWAP_{(m-1)}) \\
&+ max(P_{|+\rangle(m-1)}, P_{|0\rangle(m-1)}) + max(P_{|+\rangle(m-1)}, P_{|0\rangle(m-1)}, hCNOT_{(m-1)}, vCNOT_{(m-1)}) \\
&+ 2max(hSWAP_{(m-1)}, vSWAP_{(m-1)}, vCNOT_{(m-1)}, P_{|0\rangle(m-1)}) \\
&+ 2max(hSWAP_{(m-1)} + hCNOT_{(m-1)}, vCNOT_{(m-1)}) \\
&+ 2max(hSWAP_{(m-1)}, hCNOT_{(m-1)}) \\
&+ 2max(hSWAP_{(m-1)}, vSWAP_{(m-1)}, hCNOT_{(m-1)}, \mathcal{M}_{X(m-1)}) \\
&+ 2max(hSWAP_{(m-1)}, vSWAP_{(m-1)}, \mathcal{M}_{X(m-1)}) + 2vCNOT_{(m-1)} + 2\mathcal{M}_{X(m-1)}
\end{aligned} \tag{6d}$$

Fault-Tolerant Horizontal and Vertical SWAP Gate

$$ops(hSWAP_{(m)}) = 56hSWAP_{(m-1)} + \mathcal{EC}_{(m)} \tag{7a}$$

$$ops(vSWAP_{(m)}) = 42vSWAP_{(m-1)} + \mathcal{EC}_{(m)} \tag{7b}$$

$$time(hSWAP_{(m)}) = 8hSWAP_{(m-1)} + \mathcal{EC}_{(m)} \tag{7c}$$

$$time(vSWAP_{(m)}) = 6vSWAP_{(m-1)} + \mathcal{EC}_{(m)} \tag{7d}$$

Fault-Tolerant Pauli Gates and Hadamard

Fault-tolerant implementation of the X gate:

$$ops(X_{(m)}) = 7X_{(m-1)} + \mathcal{EC}_{(m)} \tag{8a}$$

$$time(X_{(m)}) = X_{(m-1)} + \mathcal{EC}_{(m)} \tag{8b}$$

The properties of the transversal gates $Y_{(m)}$, $Z_{(m)}$, $H_{(m)}$ as well as the transversal constructions $Strans$ and $Ttrans$ are obtained by substituting for X in the equations above.

Fault-Tolerant Measurements

Measurement in the X-basis:

$$ops(\mathcal{M}_{X(m)}) = 7\mathcal{M}_{X(m-1)} + \mathcal{EC}_{(m)} \tag{9a}$$

$$time(\mathcal{M}_{X(m)}) = \mathcal{EC}_{(m)} \tag{9b}$$

Measurement in the Z-basis:

$$ops(\mathcal{M}_{Z(m)}) = 7\mathcal{M}_{Z(m-1)} + \mathcal{EC}_{(m)} \tag{10a}$$

$$time(\mathcal{M}_{Z(m)}) = \mathcal{EC}_{(m)} - \mathcal{M}_{X(m-1)} + max(\mathcal{M}_{X(m-1)}, \mathcal{M}_{Z(m-1)}) \tag{10b}$$

Fault-Tolerant Preparation of a Logical State

Note that the expected number of gates required to re-initialize the ancillas upon initialization failure is negligible, and, therefore, we ignore these operations. Preparation of the logical state $|+\rangle$:

$$\begin{aligned}
&ops(P_{|+\rangle(m)}) = (2+2+3)hCNOT_{(m-1)} + 7vCNOT_{(m-1)} \\
&\quad + 7H_{(m-1)} + 9hSWAP_{(m-1)} + 11vSWAP_{(m-1)} \\
&\quad + 4P_{|+\rangle(m-1)} + 6P_{|0\rangle(m-1)} + 3\mathcal{M}_{X(m-1)} + \mathcal{EC}_{(m)}
\end{aligned} \tag{11a}$$

$$\begin{aligned}
&time(P_{|+\rangle(m)}) = max(P_{|+\rangle(m-1)}, P_{|0\rangle(m-1)}) \\
&+ max(P_{|+\rangle(m-1)}, P_{|0\rangle(m-1)}, hCNOT_{(m-1)}, vCNOT_{(m-1)}) \\
&+ max(hSWAP_{(m-1)}, vSWAP_{(m-1)}, vCNOT_{(m-1)}, P_{|0\rangle(m-1)}) \\
&+ max(hSWAP_{(m-1)} + hCNOT_{(m-1)}, vCNOT_{(m-1)}) + max(hSWAP_{(m-1)}, hCNOT_{(m-1)}) \\
&+ max(hSWAP_{(m-1)}, vSWAP_{(m-1)}, hCNOT_{(m-1)}, \mathcal{M}_{X(m-1)}) \\
&+ max(hSWAP_{(m-1)}, vSWAP_{(m-1)}, \mathcal{M}_{X(m-1)}) + \mathcal{EC}_{(m)} \\
&- max(P_{|+\rangle(m-1)}, P_{|0\rangle(m-1)}) + max(P_{|+\rangle(m-1)}), P_{|0\rangle(m-1)}, vSWAP_{(m-1)}) \\
&- max(P_{|+\rangle(m-1)}, P_{|0\rangle(m-1)}, hCNOT_{(m-1)}, vCNOT_{(m-1)}) \\
&+ max(P_{|+\rangle(m-1)}, P_{|0\rangle(m-1)}, hCNOT_{(m-1)}, vCNOT_{(m-1)}, H_{(m-1)})
\end{aligned} \tag{11b}$$

Preparation of the logical state $|0\rangle$:

$$\begin{aligned}
&ops(P_{|0\rangle(m)}) = (2+2+3)hCNOT_{(m-1)} + 7vCNOT_{(m-1)} + 9hSWAP_{(m-1)} \\
&\quad + 11vSWAP_{(m-1)} + 4P_{|+\rangle(m-1)} + 6P_{|0\rangle(m-1)} + 3\mathcal{M}_{X(m-1)} + \mathcal{EC}_{(m)}
\end{aligned} \tag{12a}$$

$$\begin{aligned}
&time(P_{|0\rangle(m)}) = max(P_{|+\rangle(m-1)}, P_{|0\rangle(m-1)}) \\
&\quad + max(P_{|+\rangle(m-1)}, P_{|0\rangle(m-1)}, hCNOT_{(m-1)}, vCNOT_{(m-1)}) \\
&\quad + max(hSWAP_{(m-1)}, vSWAP_{(m-1)}, vCNOT_{(m-1)}, P_{|0\rangle(m-1)}) \\
&\quad + max(hSWAP_{(m-1)} + hCNOT_{(m-1)}, vCNOT_{(m-1)}) \\
&\quad + max(hSWAP_{(m-1)}, hCNOT_{(m-1)}) \\
&\quad + max(hSWAP_{(m-1)}, vSWAP_{(m-1)}, hCNOT_{(m-1)}, \mathcal{M}_{X(m-1)}) \\
&\quad + max(hSWAP_{(m-1)}, vSWAP_{(m-1)}, \mathcal{M}_{X(m-1)}) + \mathcal{EC}_{(m)} \\
&\quad - max(P_{|+\rangle(m-1)}, P_{|0\rangle(m-1)}) + max(P_{|+\rangle(m-1)}, P_{|0\rangle(m-1)}, vSWAP_{(m-1)})
\end{aligned} \tag{12b}$$

Preparation of the logical cat state:

$$\begin{aligned}
&ops(P_{cat(m)}) = 6hCNOT_{(m-1)} + 2vCNOT_{(m-1)} + 9hSWAP_{(m-1)} \\
&\quad + 8vSWAP_{(m-1)} + P_{|+\rangle(m-1)} + 7P_{|0\rangle(m-1)} + \mathcal{M}_{Z(m-1)} + \mathcal{EC}_{(m)}
\end{aligned} \tag{13a}$$

$$\begin{aligned}
&time(P_{cat(m)}) = max(P_{|+\rangle(m-1)}, P_{|0\rangle(m-1)}) + 2vCNOT_{(m-1)} \\
&\quad + 3hCNOT_{(m-1)} + max(\mathcal{M}_{Z(m-1)}, hSWAP_{(m-1)}, vSWAP_{(m-1)}) \\
&\quad + 2max(hSWAP_{(m-1)}, vSWAP_{(m-1)}) \\
&\quad + hSWAP_{(m-1)} + vSWAP_{(m-1)} + \mathcal{EC}_{(m)}
\end{aligned} \tag{13b}$$

Fault-Tolerant S Gates

$$\begin{aligned}
&ops(S_{(m)}) = P_{|0\rangle(m)} + 4Strans_{(m)} + 2P_{|cat\rangle(m)} + 14vCNOT_{(m-1)} \\
&\quad + hCNOT_{(m)} + 14\mathcal{M}_{X(m-1)} + \mathcal{M}_{Z(m)} + \mathcal{EC}_{(m)}
\end{aligned} \tag{14a}$$

$$\begin{aligned}
&time(S_{(m)}) = max((P_{|0\rangle(m)} + Strans_{(m)}), P_{|cat\rangle(m)}) + max(Strans_{(m)}, P_{|cat\rangle(m)}) \\
&\quad + 2hCNOT_{(m-1)} + hCNOT_{(m)} + 2max(Strans_{(m)}, \mathcal{M}_{X(m-1)}) + \mathcal{M}_{Z(m)} + \mathcal{EC}_{(m)}
\end{aligned} \tag{14b}$$

Fault-Tolerant T Gates

$$\begin{aligned}
&ops(T_{(m)}) = P_{|0\rangle(m)} + 4Ttrans_{(m)} + 2P_{|cat\rangle(m)} + 14vCNOT_{(m-1)} \\
&\quad + hCNOT_{(m)} + 14\mathcal{M}_{X(m-1)} + \mathcal{M}_{Z(m)} + \mathcal{EC}_{(m)}
\end{aligned} \tag{15a}$$

$$\begin{aligned}
&time(T_{(m)}) = max((P_{|0\rangle(m)} + Ttrans_{(m)}), P_{|cat\rangle(m)}) + max(Ttrans_{(m)}, P_{|cat\rangle(m)}) \\
&\quad + 2hCNOT_{(m-1)} + hCNOT_{(m)} + 2max(Ttrans_{(m)}, \mathcal{M}_{X(m-1)}) + \mathcal{M}_{Z(m)} + \mathcal{EC}_{(m)}
\end{aligned} \tag{15b}$$

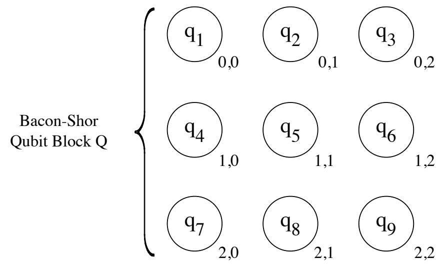


Fig. 10. 3×3 representation of the Bacon Shor code.

7 Error Correction with the Bacon-Shor Code

In this section we estimate the overhead imposed by the Bacon Shor $[[9, 3, 1]]$ error-correcting code. This section is organized as follows. First we describe basic properties of the code and the basic circuits used to implement a universal gate set using an ancilla factory model, i.e., a zone of the computer in charge of distilling high quality ancillas required to execute S and T gates. Then we build the recursive relations resulting from the concatenated code structure in order to quantify the total number of gates and total time required by each elementary operation.

7.1 The Bacon Shor Code

Logical qubits in the [[9,3,1]] Bacon Shor subsystem code are encoded using nine qubits. The Bacon Shor Code is a subsystem stabilizer code [31] better pictured in a 3×3 array with stabilizers

$$g_1 = \begin{array}{ccc} X & X & X \\ X & X & X \\ I & I & I \end{array} \qquad g_2 = \begin{array}{ccc} I & I & I \\ X & X & X \\ X & X & X \end{array}$$

$$g_3 = \begin{array}{ccc} Z & Z & I \\ Z & Z & I \\ Z & Z & I \end{array} \qquad g_4 = \begin{array}{ccc} I & Z & Z \\ I & Z & Z \\ I & Z & Z \end{array}$$

that map the encoded logical states to themselves. Logical X (or Z) Pauli operators can be executed via a homogeneous X (or Z) pulse modulo stabilizer operations, which in particular implies that logical X (or Z) operators can be implemented by a homogeneous action on one row (or column) of the array. There is no unique way of writing the codewords given the gauge freedom induced by the subsystem structure. There exists a set of gauge operators O_G consisting of pairs of X (or Z) Pauli operators acting on the same column (or row) which commute with the stabilizer and logical operators and thus act trivially on the encoded information.

The Bacon Shor code can be concatenated, each Level m encoded qubit block is built using nine Level $m-1$ logical qubits and gates. Level 1 blocks are at the lowest level of encoding and they are built of Level 0 qubits and gates provided at the physical level. Each of the gates below show how to execute an operation at some level of concatenation m using gates of level $m-1$.

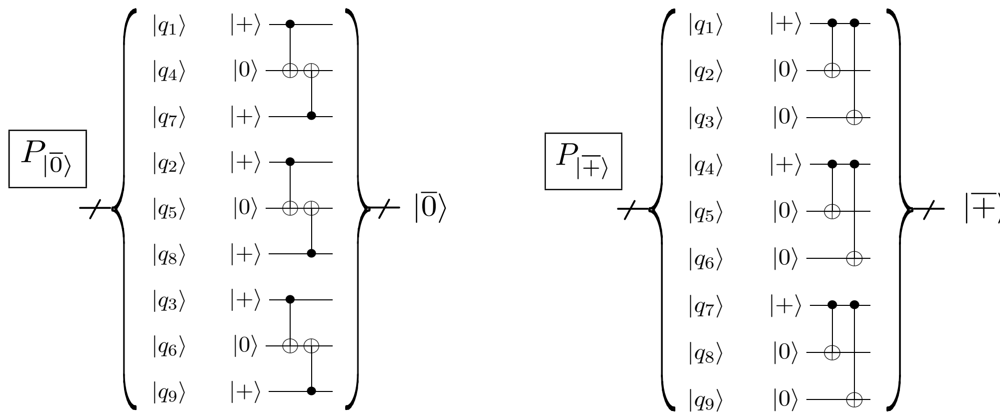


Fig. 11. Illustration of the state preparation with the Bacon-Shor code.

We will use the same notation as in the previous section. Namely, standard gates, measurement operations and state preparation at the m-th level of concatenation are denoted by $X_{(m)}$, $Y_{(m)}$, $Z_{(m)}$, $CNOT_{(m)}$, $H_{(m)}$, $S_{(m)}$, $T_{(m)}$, $\mathcal{M}_{X(m)}$, $\mathcal{M}_{Z(m)}$, $P_{|0\rangle(m)}$ and $P_{|+\rangle(m)}$. The error correction and detection is represented by $\mathcal{EC}_{(m)}$ and $\mathcal{ED}_{(m)}$. Operations required to implement a gate are denoted *ops*(...), and the required time is *time*(...).

First we consider the preparation of encoded states. The circuit shown in Figure 11 shows how to encode logical $|0\rangle$ and $|+\rangle$ at level m of concatenation, using as resources gates at level $m-1$.

Having described how to prepare encoded states, let us proceed to show how one can manipulate them, and how to implement a set of universal gates.

The Bacon-Shor code belongs to the family of Calderbank-Shor-Steane (CSS) codes, a family of codes with transversal implementation of most gates, including the $CNOT$ gate. Transversal $CNOT$ gate at Level m can be obtained by applying nine $CNOT$ gates to the corresponding control and target qubits at Level $m-1$. Figure 12 shows a fault-tolerant implementation of the $CNOT$ gate. Gates that can be performed transversally also include the single qubit Pauli gates. The H gate is transversal modulo a π rotation of the 3×3 array along the diagonal of the array, i.e. a relabelling of the qubits $q_{ij} \leftrightarrow q_{ji}$. Figure 12 also shows how to do measurement.

The S gate can be implemented using the circuit in Figure 13. It uses an ancilla in the state $|+i\rangle = \frac{|0\rangle+i|1\rangle}{\sqrt{2}}$ as a resource to generate the required gate. In turn, an encoded $|\overline{+i}\rangle$ state at any level of concatenation can be obtained via the injection circuit in Figure 14, which basically teleports an arbitrary lower-level state, in this case $|+i\rangle$, into an encoded state $|\overline{+i}\rangle$ at the cost of decoding (via $\mathcal{D}$) an encoded Bell pair. Moreover, since the injection circuit is not fault-tolerant, a higher fidelity $|\overline{+i}\rangle$ has to be distilled via multiple *successful* rounds of the circuit in Figure 16.

In order to have a universal gate set, we also need the $\pi/8$ gate called the T gate. A fault-tolerant version of this gate cannot be constructed transversally. A fault-tolerant T gate is shown in Figure 18. This gate sequence was originally constructed in [31] using one-bit

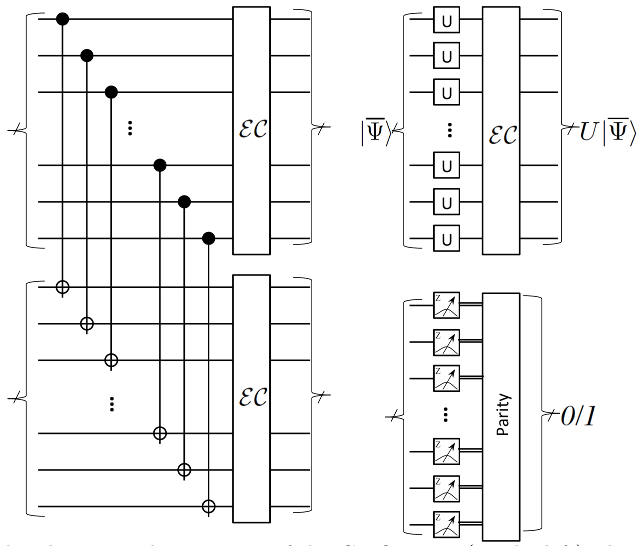


Fig. 12. Fault-tolerant implementation of the $CNOT$ gate (on the left), the single qubit gates X, Y, Z and H (top right corner), and Z basis measurement (bottom right).

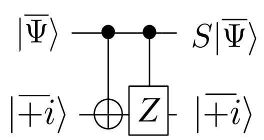


Fig. 13. Implementation of the S gate via the use of the $|+i\rangle$ resource ancilla.

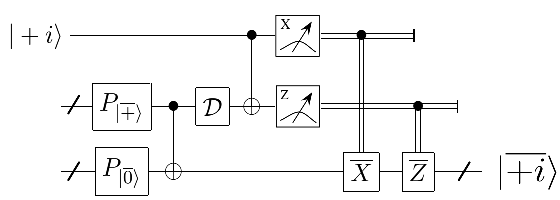


Fig. 14. Injection of an arbitrary encoded state. In this figure the injection of the $|+i\rangle$ state is shown.

teleportation. The gate sequence teleports the state $|\psi\rangle$ from the data block to the ancilla and applies the T gate to the state. The ancilla state $T\,|\overline{+}\rangle$ is prepared using the state injection method described before, followed by several *successful* rounds of the distillation circuit shown in Figures 16 and 19.

To correct errors during the computation, one uses the circuit depicted in Figure 21. This circuit corrects Z and X errors independently. In essence, the circuit extracts the measurement outcomes of the X and Z type gauge operators independently, and with various classical parity operations on the measurement outcomes it is possible to construct the recovery procedure $\mathcal{R}_X$ or $\mathcal{R}_Z$. This is detailed in Ref. [31]

Note that the error detection operation denoted $\mathcal{ED}$ differs from the error correction $\mathcal{EC}$

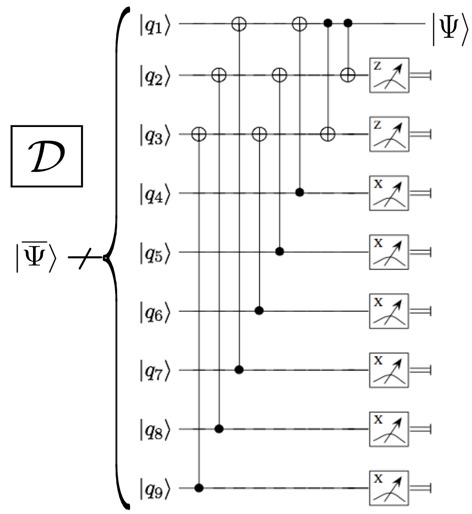


Fig. 15. Decoder circuit $\mathcal{D}$ from level s to level $s-1$. Injecting a state at level m requires a decoder from level m to level 0.

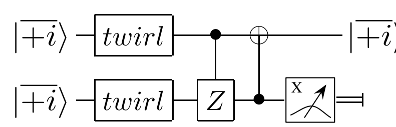


Fig. 16. Distillation process for $|+i\rangle$ states. The process is successful when the measure outcome is a 0. If it is a 1, the process must be restarted and the states discarded.

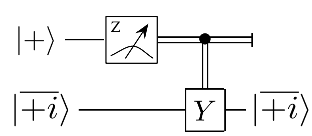


Fig. 17. Twirl gate for the $|+i\rangle$ state shown in the $|+i\rangle$ distillation circuit.

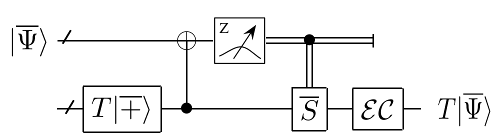


Fig. 18. Implementation of a T gate using as resource ancilla the state $T\,|+\rangle$. This state is injected with the circuit in Fig.14. The distillation and twirl procedures are different that those of the $|+i\rangle$ state.

depicted in Figure 21 by removing the two error-correction operations $\mathcal{R}_X$ and $\mathcal{R}_Z$.

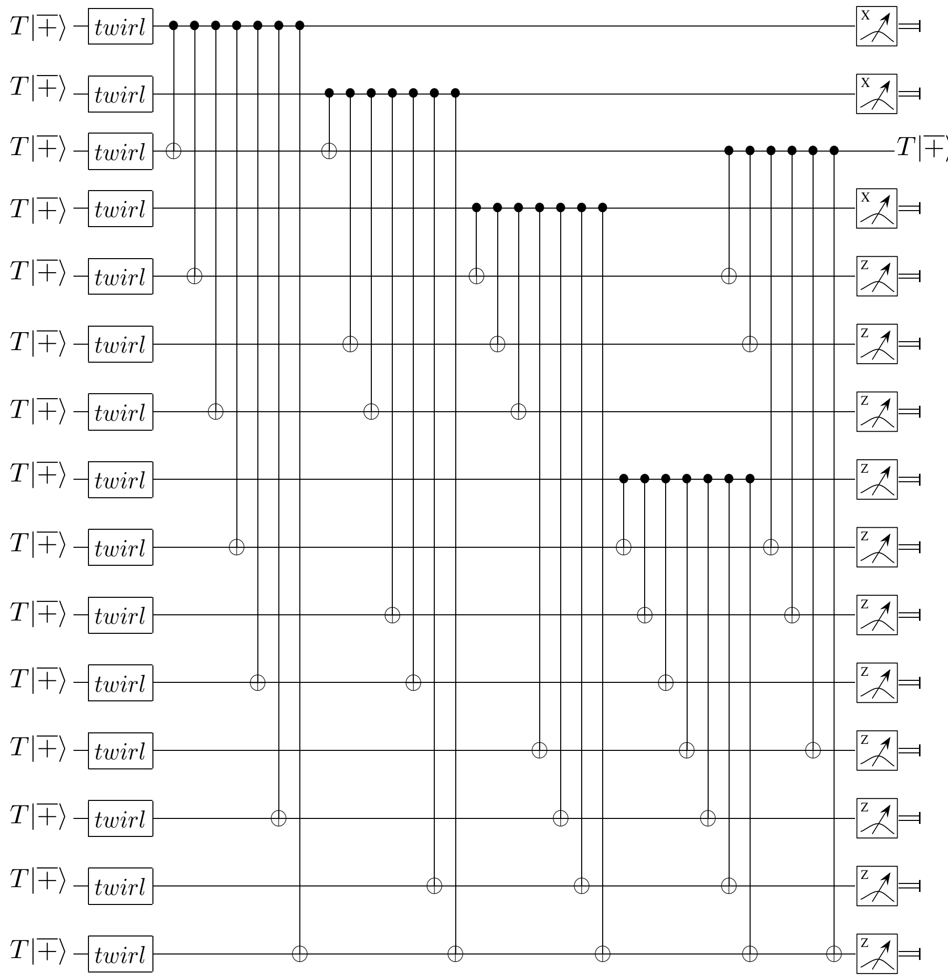


Fig. 19. Distillation circuit for the $T\,|{+}\rangle$.

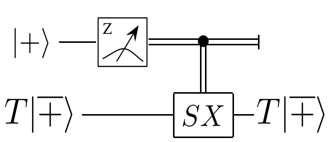


Fig. 20. Twirl operation for the $T\,|{+}\rangle$ state.

7.2 Resources with m Levels of Concatenation

Here we estimate the resources needed to perform a single logical gate at Level m of concatenation. Specifically, we calculate the number of Level 0 gates, the total number of ancillas used during the computation, and the total gate time taking parallelization into account. As a general rule we shall take into account the presence of the input and output error-correction

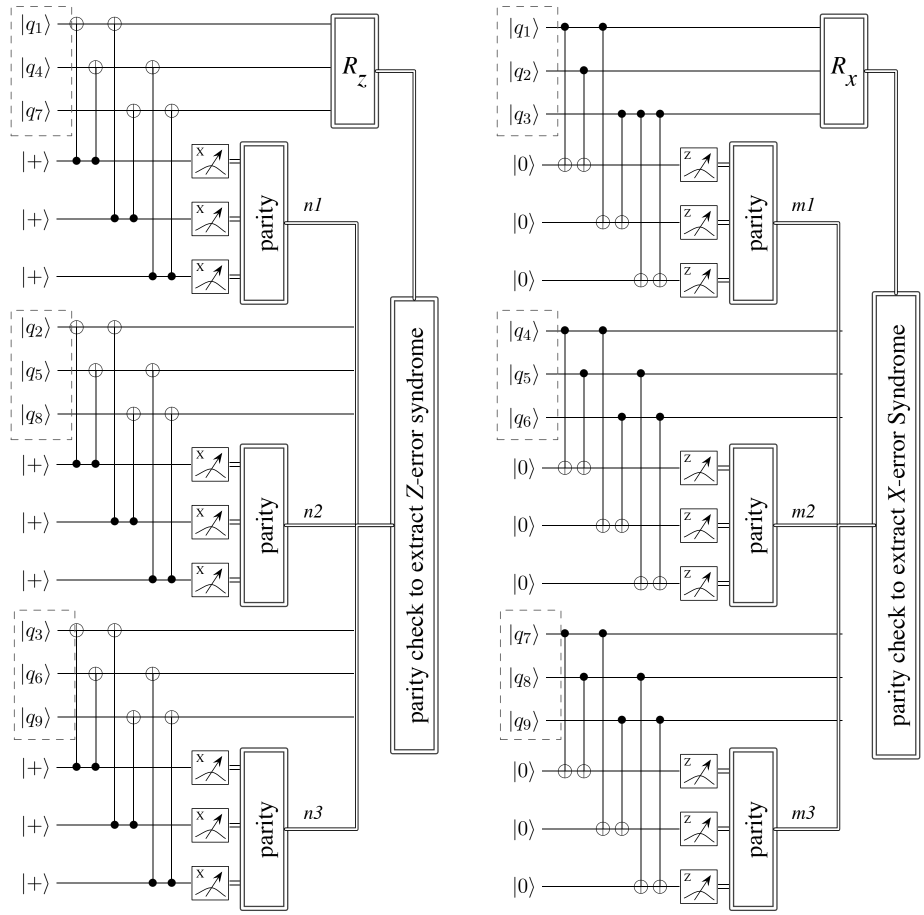


Fig. 21. Illustration of the Steane-EC syndrome extraction method.

routines inserted in any gate, and this should be understood for every circuit shown above. For example, gate $U_{(m)}$ at level m is implemented as the sequence $\mathcal{EC}_{(m)}|_{\rm inputs}U_{(m)}\mathcal{EC}_{(m)}|_{\rm outputs}$. Moreover, when one has multiple gates in action $A_{(m)}B_{(m)}$... each gate is assumed to be of the above form, i.e., $(\mathcal{EC}_{(m)}|_{\rm inputs}A_{(m)}\mathcal{EC}_{(m)}|_{\rm outputs})(\mathcal{EC}_{(m)}|_{\rm inputs}B_{(m)}\mathcal{EC}_{(m)}|_{\rm outputs})$.... Since $\mathcal{EC}$ routines acting back-to-back are only detrimental, i.e. same correction effect but using more gates, they can be contracted yielding $\mathcal{EC}_{(m)}A_{(m)}\mathcal{EC}_{(m)}B_{(m)}\mathcal{EC}_{(m)}$.... While this contraction process can be executed for every level of concatenation below the one analyzed in order to reduce the number of gates, here we opt to contract only at the analyzed level. Measurements only have input $\mathcal{EC}$ routines, while preparations typically have only output $\mathcal{EC}$ routines. Furthermore, we will account for the fact that the two-qubit interactions must be nearest-neighbor only, and our resource estimation counts the necessary SWAP gates.

7.2.1 Physical Layout of Qubits

In order to appropriately quantify the resources required, we describe the physical layout being used. We use the tile structure used in Ref. [12], designed to minimize the amount of SWAP operations used during error-correction routines, and thus preserve a high error threshold. Extra qubits are placed, labelled by a, in order to account for nearest neighbour only interactions in the following fashion

$$\begin{array}{ccccccc} a & a & a & a & a & a & a \\ a & q_{11} & a & q_{12} & a & q_{13} & a \\ a & a & a & a & a & a & a \\ a & q_{21} & a & q_{22} & a & q_{23} & a \\ a & a & a & a & a & a & a \\ a & q_{31} & a & q_{32} & a & q_{33} & a \\ a & a & a & a & a & a & a \end{array} \tag{16}$$

For details on the exact gate sequences being used we refer the reader to the original paper [12]. The threshold obtained for this layout, and the corresponding circuits implementing the error-correction subroutines, is $p_{\text{th}} = 1.3 \times 10^{-5}$.

7.2.2 Ancilla Factory

We will use what is known as the ancilla factory model, in which sections of the computer are devoted exclusively to producing and distilling the special ancillas required to implement the T and S gates at the highest level of concatenation. At this stage, we assume that all distillation rounds are successful while recognizing that the distillation process is a probabilistic process. Accounting for the probabilistic character of the distillation process can be done once the success rate and target fidelity of the distilled states are fixed, on a case by case basis. This finer analysis is not done in this report.

In order to determine how many successful distillation rounds are required, we take into account the error rate of the physical-level input state and physical gate error rates (bounded by $p^{(0)}$) to compute the output error rate after r successful rounds of distillation. The r_{max} for which the output error rate is below the error rate of Clifford gates at the highest level of concatenation L, given by the well known equation [2], $p^{(L)} = p_{\text{th}} \left(p^{(0)}/p_{\text{th}}\right)^{2^L}$, is the one we use in our simulations. We find that $r_{max} = 5$ and $r_{max} = 3$ distillation rounds are required for $T\left|+\right\rangle$ and $\left|+i\right\rangle$ ancillas respectively.

7.2.3 The Recursive Relations

We express these resources for a single logical gate at Level m:

Fault-Tolerant Measurements
Measurement in the Z-basis:

$$ops(\mathcal{M}_{Z(m)}) = 9M_{Z(m-1)} + \mathcal{EC}_{(m)} \tag{17a}$$

$$time(\mathcal{M}_{Z(m)}) = M_{Z(m-1)} + \mathcal{EC}_{(m)} \tag{17b}$$

Measurement in the X-basis:

$$ops(\mathcal{M}_{X(m)}) = 9M_{X(m-1)} + \mathcal{EC}_{(m)} \tag{18a}$$

$$time(\mathcal{M}_{X(m)}) = M_{X(m-1)} + \mathcal{EC}_{(m)} \tag{18b}$$

Fault-Tolerant $|0\rangle$ and $|+\rangle$ Preparation

Preparation of $|0\rangle$ states:

$$ops(P_{|0\rangle(m)}) = 24P_{|0\rangle(m-1)} + 12P_{|+\rangle(m-1)} + 51CNOT_{(m-1)} + \mathcal{EC}_{(m)} \tag{19a}$$

$$\begin{aligned} time(P_{|0\rangle(m)}) &= \max(P_{|0\rangle(m-1)}, P_{|+\rangle(m-1)}) + \max(P_{|0\rangle(m-1)}, CNOT_{(m-1)}) \\ &+ CNOT_{(m-1)} + \max(M_{Z(m-1)}, CNOT_{(m-1)}) + \max(P_{|0\rangle(m-1)}, SWAP_{(m-1)}) \\ &+ 2SWAP_{(m-1)} + CNOT_{(m-1)} + M_{X(m-1)} + \mathcal{EC}_{(m)} \end{aligned} \tag{19b}$$

Preparation of $|+\rangle$ states:

$$ops(P_{|+\rangle(m)}) = 24P_{|+\rangle_{(m-1)}} + 12P_{|0\rangle_{(m-1)}} + 51CNOT_{_{(m-1)}} + \mathcal{EC}_{(m)} \tag{20a}$$

$$\begin{aligned} time(P_{|+\rangle(m)}) &= \max(P_{|0\rangle(m-1)}, P_{|+\rangle(m-1)}) + \max(P_{|0\rangle(m-1)}, CNOT_{(m-1)}) \\ &+ CNOT_{(m-1)} + \max(M_{Z(m-1)}, CNOT_{(m-1)}) + \max(P_{|0\rangle(m-1)}, SWAP_{(m-1)}) \\ &+ 2SWAP_{(m-1)} + CNOT(m-1) + M_{X(m-1)} + \mathcal{EC}_{(m)} \end{aligned} \tag{20b}$$

Fault-Tolerant Pauli Gates

Fault-tolerant implementation of the X gate:

$$ops(X_{(m)}) = 3X_{(m-1)} + \mathcal{EC}_{(m)} \tag{21a}$$

$$time(X_{(m)}) = X_{(m-1)} + \mathcal{EC}_{(m)} \tag{21b}$$

Fault-tolerant implementation of the Z gate:

$$ops(Z_{(m)}) = 3Z_{(m-1)} + \mathcal{EC}_{(m)} \tag{22a}$$

$$time(Z_{(m)}) = Z_{(m-1)} + \mathcal{EC}_{(m)} \tag{22b}$$

CNOT Gates

Fault-tolerant implementation of the $CNOT$ gate (horizontal and vertical):

$$ops(CNOT_{(m)}) = 9CNOT_{(m-1)} + 144SWAP(m-1) + \mathcal{EC}_{(m)} \tag{23a}$$

$$time(X_{(m)}) = CNOT_{(m-1)} + 8SWAP_{(m-1)} + \mathcal{EC}_{(m)} \tag{23b}$$

Fault-Tolerant H Gate
Fault-tolerant implementation of the H gate:

$$ops(H_{(m)}) = 9H_{(m-1)} + 40vSWAP_{(m-1)} + 2\mathcal{EC}_{(m)} \tag{24a}$$

$$time(H_{(m)}) = H_{(m-1)} + 40SWAP_{(m-1)} + 3\mathcal{EC}_{(m)} \tag{24b}$$

SWAP Operation
The swap operations $hSWAP_{(m)}$ and $vSWAP_{(m)}$ swaps one of the 28 sub-blocks inside a single m-th level block with another of the 28 sub-blocks that is adjacent on the right/left and top/bottom respectively:

$$ops(hSWAP_{(m)}) = 144SWAP_{(m-1)} + \mathcal{EC}_{(m)} \tag{25a}$$

$$ops(vSWAP_{(m)}) = 144SWAP_{(m-1)} + \mathcal{EC}_{(m)} \tag{25b}$$

$$time(hSWAP_{(m)}) = 8SWAP_{(m-1)} + \mathcal{EC}_{(m)} \tag{25c}$$

$$time(vSWAP_{(m)}) = 8SWAP_{(m-1)} + \mathcal{EC}_{(m)} \tag{25d}$$

Error Detection and Correction with Steane-EC-type syndrome extraction
A fault-tolerant implementation of quantum error-correcting protocol for a single logical code block :

$$\begin{aligned} &ops(\mathcal{EC}_{(m)}) = 9P_{|+\rangle(m-1)} + 9P_{|0\rangle(m-1)} + 29CNOT_{(m-1)} \\ &\quad + 9M_{X(m-1)} + 9M_{Z(m-1)} + 12SWAP_{(m)} + X_{(m-1)} + Z_{(m-1)} \end{aligned} \tag{26a}$$

$$\begin{aligned} &time(\mathcal{EC}_{(m)}) = \max(P_{|+\rangle(m-1)}P_{|0\rangle(m-1)}) \\ &\quad + \max(P_{|+\rangle(m-1)}, P_{|0\rangle(m-1)}, CNOT_{(m-1)}) + \max(SWAP_{(m-1)}, CNOT_{(m-1)}) \\ &\quad + \max(SWAP_{(m-1)}, CNOT_{(m-1)}, M_{X(m-1)}) \\ &\quad + \max(hSWAP_{(m-1)}, CNOT_{(m-1)}, M_{X(m-1)}, M_{Z(m-1)}) \\ &\quad + \max(CNOT_{(m-1)}, M_{X(m-1)}, M_{Z(m-1)}) \\ &\quad + M_{X(m-1)} + \max(X_{(m-1)}, Z_{(m-1)}) \end{aligned} \tag{26b}$$

Error detection for a single logical code block:

$$\begin{aligned} &ops(\mathcal{ED}_{(m)}) = 9P_{|+\rangle(m-1)} + 9P_{|0\rangle(m-1)} + 29CNOT_{(m-1)} \\ &\quad + 9M_{X(m-1)} + 9M_{Z(m-1)} + 12SWAP_{(m)} \end{aligned} \tag{27a}$$

$$\begin{aligned} &time(\mathcal{ED}_{(m)}) = \max(P_{|+\rangle(m-1)}P_{|0\rangle(m-1)}) \\ &\quad + \max(P_{|+\rangle(m-1)}, P_{|0\rangle(m-1)}, CNOT_{(m-1)}) + \max(SWAP_{(m-1)}, CNOT_{(m-1)}) \\ &\quad + \max(SWAP_{(m-1)}, CNOT_{(m-1)}, M_{X(m-1)}) \\ &\quad + \max(SWAP_{(m-1)}, CNOT_{(m-1)}, M_{X(m-1)}, M_{Z(m-1)}) \\ &\quad + \max(CNOT_{(m-1)}, M_{X(m-1)}, M_{Z(m-1)}) + M_{X(m-1)} \end{aligned} \tag{27b}$$

In order to complete a universal gate set we are missing a fault-tolerant implementation of the S and T gates. The implementation these gates both require special resource ancillas. Let us first do the counting for the circuits implementing the gates assuming the corresponding ancilla is provided.

Fault-Tolerant S
Fault-tolerant implementation of the S gate:

$$ops(S_{(m)}) = P_{|+i\rangle(m)} + CNOT_{(m)} + CPHASE_{(m)} \tag{28a}$$

$$time(S_{(m)}) = P_{|+i\rangle(m)} + CNOT_{(m)} + CPHASE_{(m)} \tag{28b}$$

Here $CPHASE_{(m)}$ denotes the control phase gate. This two-qubit gate can be obtained conjugating a CNOT gate with a Hadamard gate acting on the target of the CNOT gate.

Fault-Tolerant T
Fault-tolerant implementation of the T gate:

$$ops(T_{(m)}) = P_{T|+\rangle,(m)} + CNOT_{(m)} + M_{Z(m)} + S_{(m)} \tag{29a}$$

$$time(T_{(m)}) = P_{T|+\rangle,(m)} + CNOT_{(m)} + M_{Z(m)} + S_{(m)} \tag{29b}$$

where $T\,|+\rangle = R_z(\pi/8)\,|+\rangle$.

Both states can be injected via the circuit in Figure 14. The gate count for such circuit is as follows:

Injection circuit for a state $|\Psi\rangle$
Implementation of the $|\Psi\rangle$ injection circuit:

$$\begin{aligned} ops(\mathrm{Inj}_\Psi) = {} & P_{|+\rangle,(m)} + P_{|0\rangle,(m)} + P_{|\Psi\rangle,(0)} + CNOT_{(m)} \\ & + \mathcal{D}_{(m,0)} + M_{Z(0)} + M_{X(0)} + X_{(m)} + Z_{(m)} \end{aligned} \tag{30a}$$

$$\begin{aligned} time(\mathrm{Inj}_\Psi) = {} & \max(P_{|+\rangle,(m)}, P_{|0\rangle,(m)}, P_{|\Psi\rangle,(0)}) + CNOT_{(m)} + \mathcal{D}_{(m,0)} \\ & + CNOT_{(0)} + \max(M_{Z(0)}, M_{X(0)}) + X_{(m)} + Z_{(m)} \end{aligned} \tag{30b}$$

Decoder circuit $\mathcal{D}_{(m,0)}$
Such circuit can be implemented in a level-by level fashion, namely:
$\mathcal{D}_{(m,0)} = \mathcal{D}_{(1,0)} ... \mathcal{D}_{(m-1,m-2)}\mathcal{D}_{(m,m-1)}$.

Implementation of the $\mathcal{D}_{(s,s-1)}$ gate:

$$ops(\mathcal{D}_{(s,s-1)}) = 8CNOT_{(s-1)} + 2M_{Z(s-1)} + 6M_{X(s-1)} \tag{31a}$$

$$time(\mathcal{D}_{(s,s-1)}) = 3CNOT_{(s-1)} + \max(M_{Z(s-1)}, M_{X(s-1)}) \tag{31b}$$

Since the injection circuit is not fault-tolerant, one must further improve the quality of the injected states by a distillation process. Higher fidelity resource states are obtained conditioned of *successful* distillation procedures. Here we do the gate count of a single distillation step for each of the two resource states.

Distillation circuits for $|+i\rangle$ and $T\,|+\rangle$
This circuit assumes the input of noisy ancillas obtained via $\mathrm{Inj}_{|+i\rangle,(m)}$, and thus we do not count this gate as part of the circuit.

Implementation of the $\mathrm{Dist}_{|+i\rangle,(m)}$ circuit:

$$ops(\mathrm{Dist}_{|+i\rangle,(m)}) = 2\,(twirl - |+i\rangle)_{(m)} + CPHASE_{(m)} + CNOT_{(m)} + M_{X(m)} \quad (32a)$$

$$time(\mathrm{Dist}_{|+i\rangle,(m)}) = (twirl - |+i\rangle)_{(m)} + CPHASE_{(m)} + CNOT_{(m)} + M_{X(m)} \quad (32b)$$

Implementation of the $(twirl - |+i\rangle)_{(m)}$ circuit:

$$ops((twirl - |+i\rangle)_{(m)}) = M_{Z(0)} + Y_{(m)} \quad (33a)$$

$$time(\mathrm{Dist}_{|+i\rangle,(m)}) = M_{Z(m)} + Y_{(m)} \quad (33b)$$

where $Y = iXZ$.

This circuit assumes the input of noisy ancillas obtained via $\mathrm{Inj}_{T|+\rangle,(m)}$, and thus we do not count this gate as part of the circuit.

Implementation of the $\mathrm{Dist}_{T|+\rangle,(m)}$ circuit:

$$ops(\mathrm{Dist}_{T|+\rangle,(m)}) = 15\,(twirl - T\,|+\rangle)_{(m)} + 32CNOT_{(m)} + 14M_{X(m)} \quad (34a)$$

$$time(\mathrm{Dist}_{T|+\rangle,(m)}) = (twirl - T\,|+\rangle)_{(m)} + 21\,CNOT_{(m)} + M_{X(m)} \quad (34b)$$

Implementation of the $(twirl - T\,|+\rangle)_{(m)}$ circuit:

$$ops((twirl - |+i\rangle)_{(m)}) = M_{Z(0)} + S_{(m)} + X_{(m)} \quad (35a)$$

$$time(\mathrm{Dist}_{|+i\rangle,(m)}) = M_{Z(m)} + S_{(m)} + X_{(m)} \quad (35b)$$

With these recursive relations, the resource count can be carried out.

8 Error Correction with the Knill's Post-Selection Scheme

Now we analyze the resource estimates of the quantum computer using the Knill's post-selection scheme [32,33]. More details can be found in [34]. The Knill's post-selection scheme concatenates $(L-1)$ levels of an error-detecting code C_{ed} with an error-correcting code C_{ec} at the top-level. We use C_{ed}^{m} to denote the Level-m encoding of the C_{ed} code.

8.1 The Knill's Post-Selection Scheme

The error-detecting code C_{ed} is the $[4, 2, 2]$ code with a stabilizer group generated by $XXXX$ and $ZZZZ$. This quantum code encodes two logical qubits and can simultaneously detect any single qubit X error and any single qubit Z error. In the Knill's scheme, we use only one of the logical qubits and treat the other as a spectator qubit. The logical operators are

$$\begin{aligned} X^L &= XXII, \\ Z^L &= ZIZI, \\ X^S &= IXIX, \\ Z^S &= IIZZ, \end{aligned}$$

where L and S are labels for the logical and the spectator qubits, respectively. Thus

$$Y^L = iX^L Z^L = YXZI.$$

The state $\left|\overline{0}\right\rangle_L \left|\overline{+}\right\rangle_S$ and $\left|\overline{+}\right\rangle_L \left|\overline{0}\right\rangle_S$ correspond to the logical $\left|\overline{0}\right\rangle$ and logical $\left|\overline{+}\right\rangle$, respectively. These states can be fault-tolerantly prepared by the circuits in Figure 22.

To perform fault-tolerant error detection ($\mathcal{ED}$) of C_{ed}^m, the circuits in Figure 23 are used depending on the state of the spectator qubit. We choose $\mathcal{ED}_0$ or $\mathcal{ED}_+$ when the spectator qubit is $\left|\overline{+}\right\rangle_S$ or $\left|\overline{0}\right\rangle_S$, respectively. Thus the state of the spectator qubit alternates between $\left|\overline{+}\right\rangle_S$ and $\left|\overline{0}\right\rangle_S$ after each error detection block. According to [32,33], the $\mathcal{ED}_0$ gate is better suited for detecting Z errors, while the $\mathcal{ED}_+$ gate is better suited for detecting X errors. If there are no errors detected, the measurement outcomes of the the first two code blocks determines the logical Pauli operator that need to be applied to the second ancilla block to complete the teleportation (this is not shown in Figure 23).

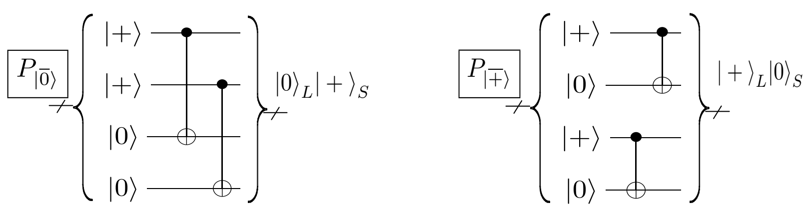


Fig. 22. Fault-tolerant preparation of the logical states $\left|\overline{0}\right\rangle$ and $\left|\overline{+}\right\rangle$.

The top-level error-correcting code C_{ec} can be the Steane code or the Bacon-Shor code, and we choose the Steane code in this work. The logical $|0\rangle$ of the concatenation of C_{ed}^m and C_{ec} is generated by the circuit in Figure 24. If an error is detected at any error detection step at any level of concatenation, the preparation of the $\left|\overline{\Phi_0}\right\rangle$ should be restarted. We use Knill's syndrome extraction method shown in Figure 25 at the top-level of concatenation.

The logical $CNOT$ gate between different code blocks can be done transversally by applying bitwise $CNOT$ gates. The $SWAP$ of qubits 2 and 3 implements the $SWAP$ of the logical qubit and the spectator qubit. We call this the *inner* $SWAP$, which is different from the *outer* $SWAP$ between two code blocks. The logical Hadamard gate is implemented by transversally applying the Hadamard gates, followed by an inner $SWAP$. The inner $SWAP$

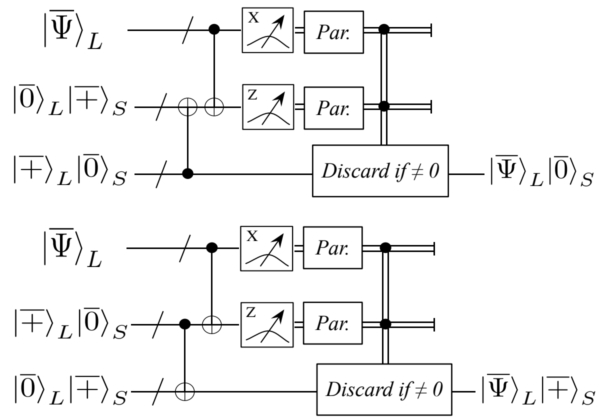


Fig. 23. Circuits for fault-tolerant quantum error detection. Top: $\mathcal{ED}_0$. Bottom: $\mathcal{ED}_+$.

does not need to be applied. Instead, we switch the labels of the qubits and keep track of them. We assume this can be done efficiently.

To achieve universal quantum computation with the Knill's post-selection scheme, it remains to prepare the the $+1$ eigenstate of Y $|\overline{+i}\rangle = \frac{1}{\sqrt{2}}\left(|\overline{0}\rangle + i\,|\overline{1}\rangle\right)$ and the magic state $T\,|\overline{+}\rangle$ at level-$(L-1)$. We use the ancilla factory of the Bacon-Shor code as described in the previous section. The only difference is the decoding operation in the injection circuit, which is shown in Figure 26 for the C_4 code.

8.2 The Qubit Layout of the C_4 Code

Herein we describe the physcial qubit layout for the Knill's post-selection scheme and estimate the number of physical gate operations and time required for each logical operation. As in the previous sections, a gate operation at level m is followed by an error-correction (detection) routine at level m.

Following the tile structures presented in [12, 35], we design a 2-dimensional 5×5 lattice architecture of physical qubits to represent a logical qubit of the C_4 code. A tile is initialized as one of the following two structures:

$$\text{Structure I}: \begin{bmatrix} O & O & O & O & O \\ O & d_1 & O & O & d_3 \\ O & O & O & O & O \\ O & O & O & O & O \\ O & d_2 & O & O & d_4 \end{bmatrix},$$

$$\text{Structure II}: \begin{bmatrix} O & O & O & O & O \\ O & O & O & O & O \\ O & O & d_1 & d_3 & O \\ O & O & d_2 & d_4 & O \\ O & O & O & O & O \end{bmatrix}.$$

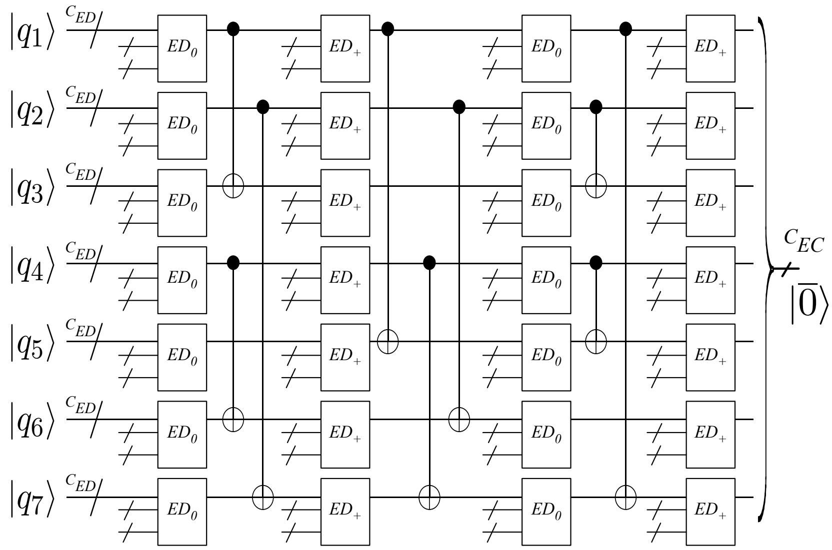


Fig. 24. Preparation of C_{ec} encoded logical state $|\overline{0}\rangle$ using C_{ed} code blocks.

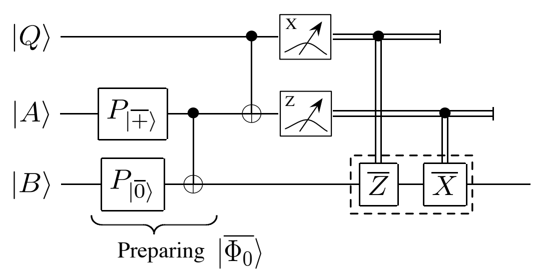


Fig. 25. Circuit for the Knill syndrome extraction.

The four data qubits of the C_4 code are denoted by d_1, d_2, d_3, and d_4. The O's are dummy qubits used for swapping with the data or ancilla qubits in communication or for the ancilla preparation and their states are irrelevant to computation. Each qubit in the tile is encoded in a lower-level tile structure.

We have the following operations of the C_4 code:

1. Error detection (ED)
2. horizontal and vertical CNOT gates (hCNOT/vCNOT)
3. horizontal and vertical SWAP gates (hSWAP/vSWAP)
4. Measurement in the X basis or the Z basis (M_X and M_Z)
5. The Pauli operators X, Y, Z, and the Hadamard gate (H)

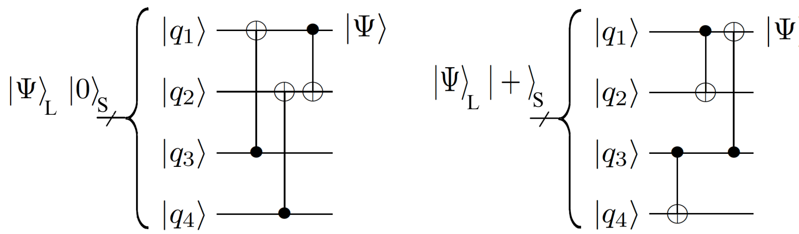


Fig. 26. The decoding circuit for C_{ed}.

6. Preparation of the ancilla qubits $|+\rangle$ or $|0\rangle$ ($P_{|+\rangle}$ and $P_{|0\rangle}$)
7. The phase gate S and the $\pi/8$ gate T

For simplicity, all lower-level gates are assumed to take one *time step*, which is the longest execution time among all gates. In reality, we may think that a qubit idles for the rest of the time step after it passes a fast gate and the error rate of this operation is the physical gate error rate plus the memory error rate for the idle time. We try to optimize the gate operations in the numbers of SWAPs, idle qubits, and time steps.

Detailed illustration of the movement and operations inside the tile is shown in Appendix 17.1. Let $ops(\text{Gate}_{(m)})$ denote the gate operation of the concatenated code at level m ($m-1$ levels of C_{ed} and one level of C_{ec}), and $ops(\text{Gate}^{C_4}_{(m)})$ denote the gate operation of the C_4 code at level m. Similarly for $time(\text{Gate}_{(m)}))$ and $time(\text{Gate}^{C_4}_{(m)}))$.

8.2.1 ErrorDetection

We first consider the error detection circuit $\mathcal{ED}_+$ in Figure 23. $\mathcal{ED}_0$ behaves in the same way.

$$\begin{aligned} ops(\mathcal{ED}^{C_4}_{(m)}) &= 4P^{C_4}_{|0\rangle(m-1)} + 4P^{C_4}_{|+\rangle(m-1)} + 6vCNOT^{C_4}_{(m-1)} + 6hCNOT^{C_4}_{(m-1)} \\ &+ 4M^{C_4}_{Z(m-1)} + 4M^{C_4}_{X(m-1)} + 4vSWAP^{C_4}_{(m-1)} + 4hSWAP^{C_4}_{(m-1)} \end{aligned} \tag{36a}$$

$$\begin{aligned} time(\mathcal{ED}^{C_4}_{(}m)) &= \max(P^{C_4}_{|0\rangle(m-1)}, P^{C_4}_{|+\rangle(m-1)}) \\ &+ \max(M^{C_4}_{X(m-1)}, M^{C_4}_{Z(m-1)}) + vCNOT^{C_4}_{(m-1)} + hCNOT^{C_4}_{(m-1)} \\ &+ \max(vCNOT^{C_4}_{(m-1)}, hCNOT^{C_4}_{(m-1)}) + vSWAP^{C_4}_{(m-1)} + hSWAP^{C_4}_{(m-1)} \end{aligned} \tag{36b}$$

We find the operation and time of $\mathcal{ED}_0$ are the same as those of $\mathcal{ED}_+$ and we omit the subscripts 0 or $+$.

8.2.2 Fault-Tolerant Horizontal and Vertical CNOT Gate

$$ops(vCNOT^{C_4}_{(m)}) = 4hCNOT^{C_4}_{(m-1)} + 8hSWAP^{C_4}_{(m-1)} + 40vSWAP^{C_4}_{(m-1)} + 2\mathcal{ED}^{C_4}_{(m)} \tag{37a}$$

$$\begin{aligned} time(vCNOT^{C_4}_{(m)}) &= hCNOT^{C_4}_{(m-1)} + 4vSWAP^{C_4}_{(m-1)} \\ &+ 2\max(vSWAP^{C_4}_{(m-1)}, hSWAP^{C_4}_{(m-1)}) + \mathcal{ED}^{C_4}_{(m)} \end{aligned} \tag{37b}$$

$$ops(hCNOT^{C_4}_{(m)}) = 4vCNOT^{C_4}_{(m-1)} + 8vSWAP^{C_4}_{(m-1)} + 40hSWAP^{C_4}_{(m-1)} + 2\mathcal{ED}^{C_4}_{(m)} \tag{37c}$$

$$\begin{aligned} time(hCNOT_{(}^{C_4}m)) = vCNOT_{(m-1)}^{C_4} + 4hSWAP_{(m-1)}^{C_4} \\ + 2\max(vSWAP_{(m-1)}^{C_4}, hSWAP_{(m-1)}^{C_4}) + \mathcal{ED}_{(m)}^{C_4} \end{aligned} \tag{37d}$$

8.2.3 Fault-Tolerant Horizontal and Vertical SWAP Gate

For a $SWAP$ operation to be fault-tolerant, we only swap a data or ancilla qubit with a dummy qubit.

Suppose $SWAP_{(0)}$ is the swap of two physical qubits on any two adjacent qubits (horizontal or vertical). Since we only swap a data or ancilla qubit with a dummy qubit, only one tile is followed by error correction.

$$ops(vSWAP_{(m)}^{C_4}) = 20vSWAP_{(m-1)}^{C_4} + \mathcal{ED}_{(m)}^{C_4} \tag{38a}$$

$$time(vSWAP_{(m)}^{C_4}) = 5vSWAP_{(m-1)}^{C_4} + \mathcal{ED}_{(m)}^{C_4} \tag{38b}$$

Similarly for $hSWAP$:

$$ops(hSWAP_{(m)}^{C_4}) = 20hSWAP_{(m-1)}^{C_4} + \mathcal{ED}_{(m)}^{C_4} \tag{39a}$$

$$time(hSWAP_{(m)}^{C_4}) = 5hSWAP_{(m-1)}^{C_4} + \mathcal{ED}_{(m)}^{C_4} \tag{39b}$$

8.2.4 Fault-Tolerant Measurements

Measurement in the X-basis:

$$ops(\mathcal{M}_{X(m)}^{C_4}) = 4M_{X(m-1)}^{C_4} + \mathcal{ED}_{(m)}^{C_4} = 4^m M_{X(0)}^{C_4} + \mathcal{ED}_{(m)}^{C_4} \tag{40a}$$

$$time(\mathcal{M}_{X(m)}^{C_4}) = M_{X(0)}^{C_4} + \mathcal{ED}_{(m)}^{C_4} \tag{40b}$$

Similarly for measurement in the Z-basis:

$$ops(\mathcal{M}_{Z(m)}^{C_4}) = 4M_{Z(m-1)}^{C_4} = 4^m M_{Z(0)}^{C_4} + \mathcal{ED}_{(m)}^{C_4} \tag{41a}$$

$$time(\mathcal{M}_{Z(m)}^{C_4}) = M_{Z(0)}^{C_4} + \mathcal{ED}_{(m)}^{C_4} \tag{41b}$$

8.2.5 Fault-Tolerant Pauli Gates and Hadamard

Fault-tolerant implementation of the X gate:

$$ops(X_{(m)}^{C_4}) = 2X_{(m-1)}^{C_4} + \mathcal{ED}_{(m)}^{C_4} = 2^m X_{(0)}^{C_4} + \mathcal{ED}_{(m)}^{C_4} \tag{42a}$$

$$time(X_{(m)}^{C_4}) = X_{(0)}^{C_4} + \mathcal{ED}_{(m)}^{C_4} \tag{42b}$$

Fault-tolerant implementation of the Z gate:

$$ops(Z_{(m)}^{C_4}) = 2Z_{(m-1)}^{C_4} + \mathcal{ED}_{(m)}^{C_4} = 2^m Z_{(0)}^{C_4} + \mathcal{ED}_{(m)}^{C_4} \tag{43a}$$

$$time(Z^{C_4}_{(m)}) = Z^{C_4}_{(0)} + \mathcal{ED}^{C_4}_{(m)} \tag{43b}$$

Fault-tolerant implementation of the Y gate:

$$\begin{aligned} ops(Y^{C_4}_{(m)}) &= X^{C_4}_{(m-1)} + Z^{C_4}_{(m-1)} + Y^{C_4}_{(m-1)} + \mathcal{ED}^{C_4}_{(m)} \\ &= 2^{m-1}(X^{C_4}_{(0)} + Z^{C_4}_{(0)}) + Y^{C_4}_{(m-1)} + \mathcal{ED}^{C_4}_{(m)} \\ &= (2^m - 1)\,(X^{C_4}_{(0)} + Z^{C_4}_{(0)}) + Y^{C_4}_{(0)} + \mathcal{ED}^{C_4}_{(m)} \end{aligned} \tag{44a}$$

$$time(Y^{C_4}_{(m)}) = Y^{C_4}_{(0)} + \mathcal{ED}^{C_4}_{(m)} \tag{44b}$$

Fault-tolerant implementation of the H gate:

$$ops(H^{C_4}_{(m)}) = 4H^{C_4}_{(m-1)} + \mathcal{ED}^{C_4}_{(m)} = 4^m H^{C_4}_{(0)} + \mathcal{ED}^{C_4}_{(m)} \tag{45a}$$

$$time(H^{C_4}_{(m)}) = H^{C_4}_{(0)} + \mathcal{ED}^{C_4}_{(m)} \tag{45b}$$

8.2.6 *Fault-Tolerant Preparation of Logical States $P_{|0\rangle} and P_{|+\rangle}$*

$$\begin{aligned} &ops(P^{C_4}_{|0\rangle(m)}) = 2P^{C_4}_{|0\rangle(m-1)} + 2P^{C_4}_{|+\rangle(m-1)} + 2hCNOT^{C_4}_{(m-1)} \\ &+ 4hSWAP^{C_4}_{(m-1)} + \mathcal{ED}^{C_4}_{(m)} \end{aligned} \tag{46a}$$

$$\begin{aligned} &time(P^{C_4}_{|0\rangle(m)}) = max(P^{C_4}_{|+\rangle(m-1)}, P^{C_4}_{|0\rangle(m-1)}) + hCNOT^{C_4}_{(m-1)} \\ &+ hSWAP^{C_4}_{(m-1)} + \mathcal{ED}^{C_4}_{(m)} \end{aligned} \tag{46b}$$

$$\begin{aligned} &ops(P^{C_4}_{|+\rangle(m)}) = 2P^{C_4}_{|0\rangle(m-1)} + 2P^{C_4}_{|+\rangle(m-1)} + 2vCNOT^{C_4}_{(m-1)} \\ &+ 4vSWAP^{C_4}_{(m-1)} + \mathcal{ED}^{C_4}_{(m)} \end{aligned} \tag{47a}$$

$$\begin{aligned} &time(P^{C_4}_{|+\rangle(m)}) = max(P^{C_4}_{|+\rangle(m-1)}, P^{C_4}_{|0\rangle(m-1)}) + vCNOT^{C_4}_{(m-1)} \\ &+ vSWAP^{C_4}_{(m-1)} + \mathcal{ED}^{C_4}_{(m)} \end{aligned} \tag{47b}$$

8.2.7 *Fault-Tolerant S and T Gates*

For universal quantum computation, we have to include the implementation of S gate and T gate. These two encoded gates of the C_4 code cannot be fault-tolerantly prepared and we adopt the ancilla factory method as in the previous section. The only difference between the ancilla factories of the Bacon-Shor code and the C_4 code is the decoding circuit as shown in Figure 26. The state of the spectator qubit determines which one of the two decoding circuits is applied. We use the circuit recursively to decode a logical state at level 0 from a logical state at level m. Since only the information matters, we don't have to swap the qubits back to its original locations.

$$ops(dec^{C_4}_{|0\rangle(m)}) = dec^{C_4}_{|0\rangle(m-1)} + 2hCNOT^{C_4}_{(m-1)} + vCNOT^{C_4}_{(m-1)} + 4hSWAP^{C_4}_{(m-1)} + 2vSWAP^{C_4}_{(m-1)} \tag{48a}$$

$$time(dec^{C_4}_{|0\rangle(m)}) = dec^{C_4}_{|0\rangle(m-1)} + hCNOT^{C_4}_{(m-1)} + vCNOT^{C_4}_{(m-1)} + vSWAP^{C_4}_{(m-1)} + hSWAP^{C_4}_{(m-1)} \tag{48b}$$

$$ops(dec^{C_4}_{|+\rangle(m)}) = dec^{C_4}_{|+\rangle(m-1)} + 2vCNOT^{C_4}_{(m-1)} + hCNOT^{C_4}_{(m-1)} + 4vSWAP^{C_4}_{(m-1)} + 2hSWAP^{C_4}_{(m-1)} \tag{49a}$$

$$time(dec^{C_4}_{|+\rangle(m)}) = dec^{C_4}_{|+\rangle(m-1)} + vCNOT^{C_4}_{(m-1)} + hCNOT^{C_4}_{(m-1)} + hSWAP^{C_4}_{(m-1)} + vSWAP^{C_4}_{(m-1)} \tag{49b}$$

Now assume the $|\overline{+i}\rangle$ and $T\,|\overline{+}\rangle$ can be efficiently prepared and transported to the destination.

$$ops(S^{C_4}_{(m)}) = 4hSWAP^{C_4}_{(m-1)} + 20vSWAP^{C_4}_{(m-1)} + 8hCNOT^{C_4}_{(m-1)} + 2H^{C_4}_{(m)} \tag{50a}$$

$$time(S^{C_4}_{(m)}) = hSWAP^{C_4}_{(m-1)} + 5vSWAP^{C_4}_{(m-1)} + 2hCNOT^{C_4}_{(m-1)} + 2H^{C_4}_{(m)} \tag{50b}$$

Note that an error correction follows the last H gates.

$$ops(T^{C_4}_{(m)}) = 8hSWAP^{C_4}_{(m-1)} + 20vSWAP^{C_4}_{(m-1)} + 4hCNOT^{C_4}_{(m-1)} + \mathcal{M}^{C_4}_{Z(m)} + S^{C_4}_{(m)} \tag{51a}$$

$$time(T^{C_4}_{(m)}) = 2hSWAP^{C_4}_{(m-1)} + 5vSWAP^{C_4}_{(m-1)} + hCNOT^{C_4}_{(m)} + \mathcal{M}^{C_4}_{Z(m)} + S^{C_4}_{(m)} \tag{51b}$$

Note that an error correction follows the S gate.

Remark: It is possible to combine a gate operation with the error correction and save several time steps.

8.3 Resources with the Concatenation Code

Here we estimate the resources needed to perform a single logical gate in the concatenation of the Steane code with $(L-1)$ levels of the C_4 code. We apply the recursive relations of the Steane code but with the lower-level gate operations of the concatenated C_4 code.

Fault-Tolerant Measurements

Measurement in the Z-basis:

$$ops(M_{Z(L)}) = 7M^{C_4}_{Z(L-1)} + \mathcal{EC}_{(L)} \tag{52a}$$

$$time(M_{Z(L)}) = M^{C_4}_{Z(L-1)} + \mathcal{EC}_{(L)} \tag{52b}$$

Measurement in the X-basis:

$$ops(M_{X(L)}) = 7M^{C_4}_{X(L-1)} + \mathcal{EC}_{(L)} \tag{53a}$$

$$time(M_{X(L)}) = M^{C_4}_{X(L-1)} + \mathcal{EC}_{(L)} \tag{53b}$$

Fault-Tolerant Pauli Gates, H Gates
Fault-tolerant implementation of the X gate:

$$ops(X_{(L)}) = 7X^{C_4}_{(L-1)} + \mathcal{EC}_{(L)} \tag{54a}$$

$$time(X_{(L)}) = X^{C_4}_{(L-1)} + \mathcal{EC}_{(L)} \tag{54b}$$

Fault-tolerant implementation of the Z gate:

$$ops(Z_{(L)}) = 7Z^{C_4}_{(L-1)} + \mathcal{EC}_{(L)} \tag{55a}$$

$$time(Z_{(L)}) = Z^{C_4}_{(L-1)} + \mathcal{EC}_{(L)} \tag{55b}$$

Fault-tolerant implementation of the Y gate:

$$ops(Y_{(L)}) = 7Y^{C_4}_{(L-1)} + \mathcal{EC}_{(L)} \tag{56a}$$

$$time(Y_{(L)}) = Y^{C_4}_{(L-1)} + \mathcal{EC}_{(L)} \tag{56b}$$

Fault-tolerant implementation of the H gate:

$$ops(H_{(L)}) = 7H^{C_4}_{(L-1)} + \mathcal{EC}_{(L)} \tag{57a}$$

$$time(H_{(L)}) = H^{C_4}_{(L-1)} + \mathcal{EC}_{(L)} \tag{57b}$$

S and T gates

$$\begin{aligned} ops(S_{(L)}) = {} & P_{|0\rangle(L)} + 28S^{C_4}_{(L-1)} + 2P_{\mathrm{cat}(L)} + 14vCNOT^{C_4}_{(L-1)} \\ & + 7hCNOT^{C_4}_{(L-1)} + 2M_{X(L)} + M_{Z(L)} + \mathcal{EC}_{(L)} \end{aligned} \tag{58a}$$

$$\begin{aligned} time(S_{(L)}) = {} & \max((P_{|0\rangle(L)} + S^{C_4}_{(L-1)}), P_{\mathrm{cat}(L)}) + \max(S^{C_4}_{(L-1)}, P_{\mathrm{cat}(L)}) \\ & + 2vCNOT^{C_4}_{(L-1)} + hCNOT^{C_4}_{(L-1)} + 2\max(S^{C_4}_{(L-1)}, M^{C_4}_{X(L-1)}) + M_{Z(L)} + \mathcal{EC}_{(L)} \end{aligned} \tag{58b}$$

$$\begin{aligned} ops(T_{(L)}) = {} & P_{|0\rangle(L)} + 28S^{C_4}_{(L-1)} + 2P_{\mathrm{cat}(L)} + 14vCNOT^{C_4}_{(L-1)} \\ & + 7hCNOT^{C_4}_{(L-1)} + 2M_{X(L)} + M_{Z(L)} + \mathcal{EC}_{(L)} \end{aligned} \tag{58c}$$

$$\begin{aligned} time(T_{(L)}) = {} & \max((P_{|0\rangle(L)} + T^{C_4}_{(L-1)}), P_{\mathrm{cat}(L)}) + \max(T^{C_4}_{(L-1)}, P_{\mathrm{cat}(L)}) \\ & + 2hCNOT^{C_4}_{(L-1)} + hCNOT^{C_4}_{(L-1)} + 2\max(T^{C_4}_{(L-1)}, M^{C_4}_{X(L-1)}) + M_{Z(L)} + \mathcal{EC}_{(L)} \end{aligned} \tag{58d}$$

Fault-tolerant implementation of the $CNOT$ gate:

$$ops(hCNOT_{(L)}) = 7hCNOT^{C_4}_{(L-1)} + 112hSWAP^{C_4}_{(L-1)} + 14vSWAP^{C_4}_{(L-1)} + 2\mathcal{EC}_{(L)} \quad \text{(59a)}$$

$$\begin{aligned}
&time(hCNOT_{(L)}) = \max(hSWAP^{C_4}_{(L-1)}, vSWAP^{C_4}_{(L-1)}) \\
&\quad + 6hSWAP^{C_4}_{(L-1)} + vCNOT^{C_4}_{(L-1)} \\
&\quad + \max(P^{C_4}_{|+\rangle(L-1)}, P^{C_4}_{|0\rangle(L-1)}, hSWAP^{C_4}_{(L-1)}) \\
&\quad + \max(P^{C_4}_{|+\rangle(L-1)}, P^{C_4}_{|0\rangle(L-1)}, hCNOT^{C_4}_{(L-1)}, \\
&\quad vCNOT^{C_4}_{(L-1)}, hSWAP^{C_4}_{(L-1)}) + \max(P^{C_4}_{|+\rangle(L-1)}, P^{C_4}_{|0\rangle(L-1)}) \\
&\quad + \max(P^{C_4}_{|+\rangle(L-1)}, P^{C_4}_{|0\rangle(L-1)}, hCNOT^{C_4}_{(L-1)}, vCNOT^{C_4}_{(L-1)}) \\
&\quad + 2\max(hSWAP^{C_4}_{(L-1)}, vSWAP^{C_4}_{(L-1)}, vCNOT^{C_4}_{(L-1)}, P^{C_4}_{|0\rangle(L-1)}) \\
&\quad + 2\max(hSWAP^{C_4}_{(L-1)} + hCNOT^{C_4}_{(L-1)}, vCNOT^{C_4}_{(L-1)}) \\
&\quad + 2\max(hSWAP^{C_4}_{(L-1)}, hCNOT^{C_4}_{(L-1)}) \\
&\quad + 2\max(hSWAP^{C_4}_{(L-1)}, vSWAP^{C_4}_{(L-1)}, hCNOT^{C_4}_{(L-1)}, M^{C_4}_{X(L-1)}) \\
&\quad + 2\max(hSWAP^{C_4}_{(L-1)}, vSWAP^{C_4}_{(L-1)}, M^{C_4}_{X(L-1)}) \\
&\quad + 2vCNOT^{C_4}_{(L-1)} + 2M^{C_4}_{X(L-1)}
\end{aligned} \quad \text{(59b)}$$

$$ops(vCNOT_{(L)}) = 7vCNOT^{C_4}_{(L-1)} + 70vSWAP^{C_4}_{(L-1)} + 12hSWAP^{C_4}_{(L-1)} + 2\mathcal{EC}_{(L)} \quad \text{(59c)}$$

$$\begin{aligned}
&time(vCNOT_{(L)}) = \max(vSWAP^{C_4}_{(L-1)} hSWAP^{C_4}_{(L-1)}) \\
&\quad + 2\max(vSWAP^{C_4}_{(L-1)} vCNOT^{C_4}_{(L-1)}) + 4 * vSWAP^{C_4}_{(L-1)} \\
&\quad + \max(P^{C_4}_{|+\rangle(L-1)} P^{C_4}_{|0\rangle(L-1)} hSWAP^{C_4}_{(L-1)} vSWAP^{C_4}_{(L-1)}) \\
&\quad + \max(P^{C_4}_{|+\rangle(L-1)} P^{C_4}_{|0\rangle(L-1)} hCNOT^{C_4}_{(L-1)} \\
&\quad vCNOT^{C_4}_{(L-1)} hSWAP^{C_4}_{(L-1)}) + \max(P^{C_4}_{|+\rangle(L-1)} P^{C_4}_{|0\rangle(L-1)}) \\
&\quad + \max(P^{C_4}_{|+\rangle(L-1)} P^{C_4}_{|0\rangle(L-1)} hCNOT^{C_4}_{(L-1)} vCNOT^{C_4}_{(L-1)}) \\
&\quad + 2\max(hSWAP^{C_4}_{(L-1)} vSWAP^{C_4}_{(L-1)} vCNOT^{C_4}_{(L-1)} P^{C_4}_{|0\rangle(L-1)}) \\
&\quad + 2\max(hSWAP^{C_4}_{(L-1)} + hCNOT^{C_4}_{(L-1)} vCNOT^{C_4}_{(L-1)}) \\
&\quad + 2\max(hSWAP^{C_4}_{(L-1)} hCNOT^{C_4}_{(L-1)}) \\
&\quad + 2\max(hSWAP^{C_4}_{(L-1)} vSWAP^{C_4}_{(L-1)} hCNOT^{C_4}_{(L-1)} M^{C_4}_{X(L-1)}) \\
&\quad + 2\max(hSWAP^{C_4}_{(L-1)} vSWAP^{C_4}_{(L-1)} M^{C_4}_{X(L-1)}) \\
&\quad + 2vCNOT^{C_4}_{(L-1)} + 2M^{C_4}_{X(L-1)}
\end{aligned} \quad \text{(59d)}$$

SWAP Operation

We swap two logical qubits at the top level of the Steane code and thus we need two error-correction blocks at this level. Notice that we do not swap a data or ancilla qubit with a dummy qubit at a lower level.

$$ops(hSWAP_{(L)}) = 112hSWAP^{C_4}_{(L-1)} + 14vSWAP^{C_4}_{(L-1)} + 2\mathcal{EC}_{(L)} \tag{60a}$$

$$time(hSWAP_{(L)}) = 8hSWAP^{C_4}_{(L-1)} + 2vSWAP^{C_4}_{(L-1)} + \mathcal{EC}_{(L)} \tag{60b}$$

$$ops(vSWAP_{(L)}) = 84vSWAP^{C_4}_{(L-1)} + 14hSWAP^{C_4}_{(L-1)} + 2\mathcal{EC}_{(L)} \tag{60c}$$

$$time(vSWAP_{(L)}) = 6vSWAP^{C_4}_{(L-1)} + 2hSWAP^{C_4}_{(L-1)} + \mathcal{EC}_{(L)} \tag{60d}$$

Fault-Tolerant Preparation of a Logical State
The detailed tile operations are given in Appendix 18.

$$\begin{aligned} ops(P_{|0\rangle(L)}) &= 4P^{C_4}_{|0\rangle(L-1)} + 3P^{C_4}_{|+\rangle(L-1)} + 4hCNOT^{C_4}_{(L-1)} \\ &+ 5vCNOT^{C_4}_{(L-1)} + 9hSWAP^{C_4}_{(L-1)} + 13vSWAP^{C_4}_{(L-1)} \end{aligned} \tag{61a}$$

$$\begin{aligned} time(P_{|0\rangle(L)}) &= \max(P^{C_4}_{|+\rangle(L-1)}, P^{C_4}_{|0\rangle(L-1)}) \\ &+ \max(hCNOT^{C_4}_{|+\rangle(L-1)}, vCNOT^{C_4}_{|0\rangle(L-1)}) + hCNOT^{C_4}_{(L-1)} \\ &+ 2\max(vCNOT^{C_4}_{|+\rangle(L-1)}, hSWAP^{C_4}_{|0\rangle(L-1)}, vSWAP^{C_4}_{|0\rangle(L-1)}) \\ &+ 3\max(hSWAP^{C_4}_{|0\rangle(L-1)}, vSWAP^{C_4}_{|0\rangle(L-1)}) + hSWAP^{C_4}_{|+\rangle(L-1)} \end{aligned} \tag{61b}$$

The tile operations of preparation of the logical state $|+\rangle$ are omitted.

$$\begin{aligned} ops(P_{|+\rangle(L)}) &= 3P^{C_4}_{|0\rangle(L-1)} + 4P^{C_4}_{|+\rangle(L-1)} + 4hCNOT^{C_4}_{(L-1)} \\ &+ 5vCNOT^{C_4}_{(L-1)} + 9hSWAP^{C_4}_{(L-1)} + 13vSWAP^{C_4}_{(L-1)} \end{aligned} \tag{62a}$$

$$\begin{aligned} time(P_{|+\rangle(L)}) &= \max(P^{C_4}_{|+\rangle(L-1)}, P^{C_4}_{|0\rangle(L-1)}) \\ &+ \max(hCNOT^{C_4}_{|+\rangle(L-1)}, vCNOT^{C_4}_{|0\rangle(L-1)}) + hCNOT^{C_4}_{(L-1)} \\ &+ 2\max(vCNOT^{C_4}_{|+\rangle(L-1)}, hSWAP^{C_4}_{|0\rangle(L-1)}, vSWAP^{C_4}_{|0\rangle(L-1)}) \\ &+ 3\max(hSWAP^{C_4}_{|0\rangle(L-1)}, vSWAP^{C_4}_{|0\rangle(L-1)}) + hSWAP^{C_4}_{|+\rangle(L-1)} \end{aligned} \tag{62b}$$

Error Detection and Correction

We use the Knill syndrome extraction in Figure 25 instead of the Steane syndrome extraction used in the Steane code. Here we assume the data tile $|Q\rangle$ and the two ancilla tiles $|A\rangle$, $|B\rangle$ are vertically arranged in the order $|Q\rangle, |A\rangle, |B\rangle$.

$$\begin{aligned} ops(EC_{(L)}) &= P_{|+\rangle(L)} + P_{|0\rangle(L)} + 14vCNOT^{C_4}_{(L-1)} + 7M^{C_4}_{Z(L-1)} \\ &+ 7M^{C_4}_{X(L-1)} + 7X^{C_4}_{L-1} + 7Z^{C_4}_{L-1} + 14vSWAP^{C_4}_{(L-1)} \end{aligned} \tag{63a}$$

$$\begin{aligned} time(EC_{(L)}) &= \max(P_{|+\rangle(L)}, P_{|0\rangle(L)}) + 2vCNOT^{C_4}_{(L-1)} \\ &+ \max(M^{C_4}_{Z(L-1)}, M^{C_4}_{X(L-1)}) + X^{C_4}_{L-1} + Z^{C_4}_{L} + 2vSWAP^{C_4}_{(L-1)} \end{aligned} \tag{63b}$$

Remark: It would be good if we can squeeze the size of the three tiles for error correction.

8.4 Error Threshold

Herein we estimate the error threshold of the Knill's post-selection scheme in the 2-dimensional tile for the case of stochastic, adversarial noise. Following the procedure presented in [11, 12, 32], we count the number of *malignant* pairs of locations in the *extended rectangle* of the $CNOT$ gate. A rectangle of the $CNOT$ gate consists of the bitwise CNOTs on two logical qubits followed by two error detection $\mathcal{ED}$ blocks. An extended rectangle of the $CNOT$ gate is a rectangle of the $CNOT$ gate with two preceding error detection blocks. As shown in Figure 27, we assume the preceding $\mathcal{ED}$s are $\mathcal{ED}_+$s and the following $\mathcal{ED}$s are $\mathcal{ED}_0$s.

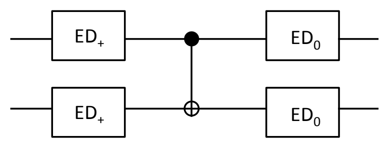


Fig. 27. The extended rectangle of the $CNOT$ gate.

A set of locations is called malignant if errors in these locations could make the calculation of the rectangle incorrect. There are seven types of locations in a $CNOT$ extended rectangle: (1) $P_{|+\rangle}$; (2) $P_{|0\rangle}$; (3) M_X; (4) M_Z; (5) $hSWAP/vSWAP$; (6) $hCNOT/vCNOT$; (7) idling qubits.

To obtain a higher error threshold, we optimize the tile operations of the extended rectangle of the CNOT gate and the animation is available online. There are 196 locations in the extended rectangle of the CNOT gate: 32 idle qubits and 154 gates, in which 38 gates are SWAPs. Here we assume the error detection blocks begin before the time step that the data qubits come in and thus there are no idle qubits at time steps 1, 2, and 3 in the preceding ED. We find that the numbers of malignant pairs of locations of each kind are given by

$$\alpha = \begin{pmatrix} 4 & 8 & 8 & 0 & 0 & 32 & 16 \\ & 0 & 0 & 14 & 96 & 80 & 32 \\ & & 16 & 0 & 96 & 104 & 32 \\ & & & 16 & 96 & 112 & 32 \\ & & & & 442 & 672 & 268 \\ & & & & & 322 & 288 \\ & & & & & & 106 \end{pmatrix},$$

where $\alpha_{i,j}$ represents the number of malignant pairs at locations of types i and j.

Let $\epsilon_j^{(m)}$ be the error rates of type j at level m. For error correction to be effective, we require

$$\epsilon_6^{(m+1)} = \sum_{i \leq j} \alpha_{i,j} \epsilon_i^{(m)} \epsilon_j^{(m)} + O((\epsilon_{\max}^{(m)})^3) \leq \epsilon_6^{(m)}, \tag{64}$$

where $\alpha_{i,j}$ is the number of malignant pairs of types i and j and $\epsilon_{\max}^{m}$ is the maximum of the seven types of error rate.

Remark: in general the error rate of a $SWAP$ gate is higher than a $CNOT$ gate since it is implemented by a series of gate operations, such as three $CNOT$ gates. However, in the 2-dimension tile, we only swap a data or ancilla qubit with a dummy qubit and the cost of such a $SWAP$ gate is less than a $CNOT$ gate as can be seen in Appendix 17.1.

We assume all errors of weight 3 or larger are malignant and the effect of errors of weight higher than three can be ignored. (This might still be an over estimate of higher-order terms.)

Let the ratio of the memory error rate of the idle qubits to the gate error rate as γ. We define the number of effective malignant pairs as

$$\sum_{i,j,=1,i>j}^{6} \alpha_{i,j} + \sum_{i=1}^{6} \gamma \alpha_{i,7} + \gamma^2 \alpha_{7,7}.$$

If we assume the error rates are the same for all types of locations, the total number of malignant pairs is 2892 and Eq. (64) becomes

$$\binom{196}{3} (\epsilon_6^{(m)})^3 + 2892 (\epsilon_6^{(m)})^2 < \epsilon_6^{(m)},$$

which gives an error threshold

$$\epsilon(\gamma = 1) < 3.06 \times 10^{-4}.$$

If we assume $\gamma = 0.1$ or $\gamma = 0$, the number of effective malignant pairs are 2185.9 and 2118.0, and the error thresholds become

$$\epsilon(\gamma = 0.1) < 4.06 \times 10^{-4},$$

and

$$\epsilon(\gamma = 0) < 4.14 \times 10^{-4},$$

respectively.

8.5 Ancilla Factory

We first analyze the ancilla factories for $\overline{|+i\rangle}$ states. The analysis of the ancilla factories for the magic state $T\overline{|+\rangle}$ follows. Suppose $\epsilon_{anc}^{(r)}$ is the error probability of the output state of r rounds of distillation protocol in Figure 16 without the twirling blocks. Let's assume the distillation circuit is perfect. From [32], we have

$$\epsilon_{anc}^{(r)} \leq \frac{1}{2} (2\epsilon_{anc}^{(0)})^{2^r},$$

where $\epsilon_{anc}^{(0)}$ is the initial error rate of preparing a $|+i\rangle$ state. A $|+i\rangle$ state can be obtained by preparing a $|+\rangle$ state followed by a phase gate S. Thus $\epsilon_{anc}^{(0)}$ can be approximated by the sum of the error rate of preparing $|+\rangle$ and the error rate of a physical phase gate S, which is assumed to be a known property of the physical quantum technology.

Then we choose the $|+i\rangle$ with error rate $\epsilon_{anc}^{(r)}$ such that the output of the injection circuit in Figure 14 at the highest level of concatenation has an error rate smaller than the error threshold of the quantum error-correction scheme. In reality, the distillation circuit is not perfect, and we would expect more rounds of distillations than the r required.

Now we have the number of the gates for a successful preparation of a $|+i\rangle$ state, which is the number of gates in the distillation protocol plus the number of gates in the injection circuit Figure 14.

Since the cost of an ancilla factory is much higher than the cost of movement of the $\overline{|+i\rangle}$ states (roughly $O(10^4)$ based on our numerical estimates), we would like to reduce the number of ancilla factories but to move the $|\overline{+i}\rangle$ states around the quantum computer. Suppose there are K logical qubits in an algorithm. Then the size of the quantum computer is roughly $\sqrt{K}$ times the size of a tile. Then the movement distance of a $|\overline{+i}\rangle$ state is at most $\sqrt{K/2}$ times the size of a tile. Thus the movement cost is about $\sqrt{K/2}$ $SWAP$ gates at the highest level of concatenation.

Now we determine the number of ancilla factories F we need. Assume a computational step for a combination of a certain algorithm, a quantum error-correcting code, and a physical technology takes time T_{comp}. A computational step could be thought of as the period between two phase gates are applied on the same qubits. Assume a factory takes time T_{anc} to prepare a $|\overline{+i}\rangle$ state. (These ancilla qubits are prepared off-line and they are preserved and moved to some target locations by $SWAP$ gates.) Assume the maximum number of phase gates in a computational step is N. (This number is called the paralleling factor of the phase gate in the algorithm.) Assume an ancilla factory prepares a $|\overline{+i}\rangle$ state with successful probability p_{succ}. Let b be the number of $|\overline{+i}\rangle$ states successfully generated by F ancilla factories. Note that b obeys a binomial distribution with parameters F, p_{succ}, which can be approximated by a normal distribution with parameters $\mathcal{N}(Fp_{succ}, Fp_{succ}(1-p_{succ}))$. Then F ancilla factories can on average generate

$$\frac{FT_{\text{comp}} \times p_{\text{succ}}}{T_{\text{anc}}}$$

$|\overline{+i}\rangle$ states in time T_{comp}. (We can replace T_{comp} by $T_{\text{comp}} - \sqrt{K/2}time(SWAP(m))$ for higher accuracy.) We would like this number to be significantly larger than N to ensure that we have enough $|\overline{+i}\rangle$ states. It is reasonable to choose Fp_{succ} to be 5 deviations larger than $\frac{NT_{\text{anc}}}{T_{\text{comp}}}$, that is

$$Fp_{\text{succ}} \geq 5\sqrt{Fp_{\text{succ}}(1-p_{\text{succ}})} + \frac{NT_{\text{anc}}}{T_{\text{comp}}}.$$

Thus F can be determined by choosing appropriate T_{comp}.

We obtain this estimate for two reasons: first, to avoid the cost in error correction to preserve the $|\overline{+i}\rangle$ states; and second, to guarantee that we have sufficient number of $|\overline{+i}\rangle$ states when they are needed.

9 Error Correction with the Surface Code

The rest of this document describes the methodology we utilized to estimate the resources necessary to fault tolerantly implement a given algorithm using the surface code [36]. We also refer the reader to [37] which illustrates how to do computation with the surface code. This document quantifies the cost of this computation. As before, we assume that we know the number of logical gates required to implement each algorithm, as well as statistics on gate reliability and operation time associated with the physical quantum technology that the algorithm is to be implemented on. Our high level approach is as follows. First we estimate the number of physical qubits required to perform the computation. Then we estimate the running time and finally the number of gates of each type required by the computation.

To estimate the number of physical qubits, we first establish the code distance. Given the number of logical gates in the algorithm, the threshold [38, 39] of the code and error properties of the physical quantum technology, we choose a sufficiently high code distance to guarantee that the calculation successfully finishes with probability at least 50%. We describe a tiled layout that ensures that the weight of any logical operator is equal at least to the code distance. Since the optimal layout and the number of physical qubits depends on the syndrome extraction method of choice (Steane, Shor, Knill), we describe a separate layout for each of these methodologies. In addition to reserving one tile per logical qubit required by the algorithm, we also estimate the additional space needed for ancilla distillation as well as ancillary space required to perform certain operations, such as $CNOT$.

The running time is obtained by calculating the time needed to perform each elementary operation. First, we estimate the time needed to perform one round of syndrome extraction and error correction for each of the syndrome extraction schemes. Then we estimate the time needed to perform elementary logical gates such as logical state preparation, measurement and the $CNOT$ gate. We build our estimate from the ground up, starting with estimates for the simpler logical gates which we use as building blocks in more advanced operations such as ancilla distillation.

A gate count that provides the number of gates of each type that need to be executed is obtained in two steps as follows. In the first step, we estimate the number of gates needed to perform error correction during the entire duration of the experiment. Since error correction is performed continuously, and all other operations require a small number of gates, this is the dominant factor in the final gate count. Note that the gate count critically depends on the syndrome extraction method of choice. In the second step, we add the small number of gates required to perform logical operations. These additional operations in the second step do not include syndrome measurement, and therefore the estimate can be done independently of the syndrome extraction method of choice. Also note that certain operations such as Pauli's do not require any additional physical operations as long as we keep track of the Pauli frame.

The rest of the document is organized as follows. In Section 10 we discuss the inputs (properties of quantum technologies and algorithms) needed to perform this estimate. Then in Section 11 we estimate the number of physical qubits required by the system. In Section 12 we describe the methodology used to estimate the running time on the given algorithm and physical quantum technology, and finally in Section 13 we estimate the number of gates of each type.

10 Surface Code Estimate Inputs and Goals

We want to estimate the resources required to implement an algorithm on a given physical quantum technology. For convenience we will assume that this data is given to us in the form of two data structures (Algorithm and Technology) containing numbers.

The structure Algorithm has the following fields:

1. Logical qubit count: numQubits
2. Gate counts: numPrep $|0\rangle$, numPrep $|+\rangle$, numH, numS, numS†, numT, numT†, numCNOT, etc.
3. Gate parallelism: paralPrep $|0\rangle$, paralPrep $|+\rangle$, paralH, etc.

The structure Technology has the following fields:

1. Physical gate error rate: errorRate

2. Physical gate times: Prep $|0\rangle$ Time, Prep $|+\rangle$ Time, HTime, STime, $\mathrm{S}^{\dagger}$Time, TTime, $\mathrm{T}^{\dagger}$Time, CNOTTime, XTime, etc.

Our goal is to transform these inputs into:

1. **Number of qubits:** The physical area used to implement algorithm, i.e., the number of physical qubits.

2. **Runtime:** The actual time to run algorithm in units of time.

3. **Gate count:** The number of gates of each type needed during the entire computation.

To do this, we will use the following steps:

1. **Find code distance** d**:** For a required logical gate reliability and the given physical machine-level reliabilities, determine the required size and spacing of the holes in the surface code lattice to ensure that logical operations in the code have weight at least d.

2. **Determine magic state distillation precision:** From the physical machine-level parameters, determine the approximate error in the undistilled state and the concatenation level required to distill the states to be usable in the logical circuit.

3. **Determine number of qubits:** Determine the total number of logical qubits required to support the computation, based off of the number of logical qubits in the algorithm and the number of logical ancillary qubits to support CNOT, H, S, and T operations (including distillation). Based upon a layout strategy for logical qubits, this will provide us with an estimate of the area required to implement the computation. The layout strategy used will depend upon the constraints of the physical quantum technology. We will assume that physical qubit layout is restricted to two dimensions.

4. **Determine timesteps:** Determine the total number of timesteps required to implement the algorithm, based upon the number of timesteps required for each of the logical operations in the algorithm. The number of timesteps depends on the level of parallelization that can be assumed between logical operations. From timesteps we can attain the runtime of the algorithm.

5. **Determine number of gates for error correction:** Count the number of gates needed to extract all syndromes in the entire system and to correct errors.

6. **Determine number of gates for all other operations:** Count the number of additional gates needed to perform each logical operation, multiplied by the number of occurrences of these logical operations in the algorithm.

7. **Determine the final gate count:** Add the number of gates needed by error correction and other operations.

11 Area Estimate

Here we estimate the number of physical qubits, i. e., number of physical qubits needed to encode a logical qubit, the space needed to store ancillas and distill magic states. To do this, we estimate a sufficient code distance. Then we specify a physical qubit layout for several possible syndrome extraction techniques and provide the final pysical qubit estimate.

11.1 Background and Terminology

Here we discuss some surface code terminology necessary to explain the process behind our estimate.

The surface code [36] places qubits on a **lattice** such as the one shown in Figure 28 a). The data qubits are located on each edge of the grid.

The code has **stabilizers** of type $XXXX$ and $ZZZZ$. The $XXXX$ stabilizers correspond to the orange diamonds in Figure 28 a), and are also called site stabilizers. The $ZZZZ$ stabilizers correspond to the blue diamonds and are called plaquette stabilizers. Note that stabilizers of weight three shown as triangles may be present at the edges of the lattice (we assume absence of periodic boundary conditions), or on the perimeter of holes in the surface.

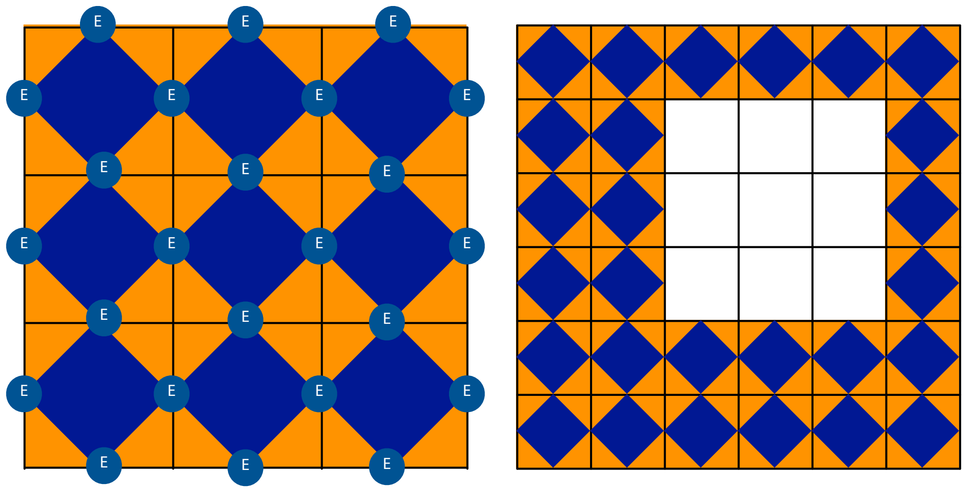
To measure each of the stabilizers, we require one or more **ancilla** qubits. The syndrome measurement circuits, required number of ancillas, and optimal qubit layout differs for each syndrome extraction method. Detailed analysis for the Steane, Shor, and Knill syndrome extraction method is shown in Section 11.4.1, 11.4.2, and 11.4.3, respectively. Initially, to facilitate the analysis before discussing specifics of the three syndrome extraction methods, we will think of the lattice in terms of the grid in Figure 28. Note that if we have a grid that is w squares wide and h squares tall, we will have a surface consisting of $2wh + w + h$ physical qubits (excluding syndrome measurement ancillas).

As discussed in [37] logical qubits are represented by areas in the surface where stabilizers are not enforced. These areas are called **holes**. Holes always come in pairs. If we don't enforce a rectangular region of the grid cells, we create a "smooth" hole. Figure 28 b) shows a smooth hole on the grid. If we don't enforce a region of stabilizers that is shifted by half a grid cell (in both spatial dimensions), we create a "rough" hole. Since logical Pauli operators in the code are loops that wrap around a hole or connect pairs of holes, the size and minimum spacing of these holes affects the fault tolerance of the system. The larger the hole, the better the fault tolerance.

The length of the logical operator with the smallest weight is called **code distance**. In the next subsection, we describe how to choose code distance to ensure that the computation finishes with the correct result with high probability.

11.2 Layout with Sufficient Code Distance

Let each logical qubit be represented by a pair of smooth holes in the surface. These holes are positioned on the grid so that the distance between two holes is at least d squares (the code distance) to maintain fault-tolerance. Figure 29 shows this layout. Since each logical qubit may be involved in a braiding operation later on (where another hole will be in between the



(a) $XXXX$ site stabilizers are shown in orange and $ZZZZ$ plaquette stabilizers in blue

(b) A hole in the surface

Fig. 28: The lattice of the surface code.

two holes representing a logical qubit), we set the distance of the two holes in the logical qubit to be a distance of $3d$ away from each other by default. This will allow us to place a hole of width d in the area between the two holes while still maintaining a distance of d between all three holes. This thereby facilitates braiding operations between logical qubits. We also lay out each pair of holes next to each other, maintaining a distance of $3d$ between the edges of every hole pair, allowing enough room for hole movement to occur between different logical qubits.

The required code distance d can be estimated by solving the following equation (as shown in [40])

$$\epsilon_{Logical} \geq C_1 \left(C_2 \frac{\epsilon_{Physical}}{\epsilon_{Threshold}} \right)^{\lfloor \frac{d+1}{2} \rfloor}. \tag{65}$$

This equation involves the following variables:

- **Logical error rate** $\epsilon_{Logical}$**:** The required logical error rate to guarantee high probability of success can be estimated as $\epsilon_{Logical} \approx 0.5/Algorithm.numGates$.
- **Physical error rate** $\epsilon_{Physical}$**:** The physical error rate depends on the parameters of the physical technology (here, $\epsilon_{Physical} =$ Technology.errorRate).
- **Code threshold** $\epsilon_{Threshold}$**:** Several estimates of the threshold of the surface code have been presented in literature. Here, we use the numerical estimate $\epsilon_{Threshold} \approx 0.01$ from [37].
- **Code constants** C_1 **and** C_2: The constants depend on the properties of the code. As has been done in [40], we use the estimate $C_1 \approx 0.13$ and $C_2 \approx 0.61$.

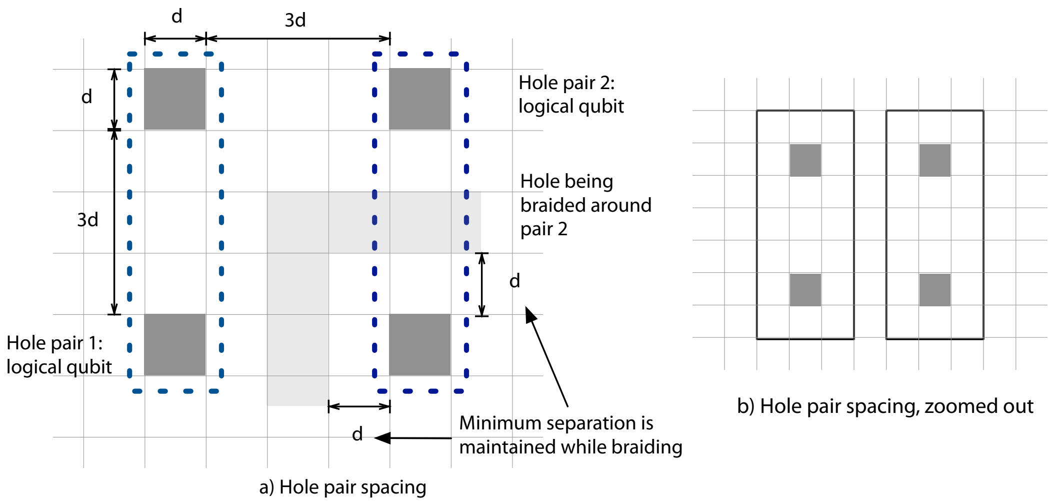


Fig. 29: Spacing between holes representing logical qubits.

11.3 Ancilla Space

To store certain number of logical qubits, we need the same number of hole pairs. However, to actually perform computation, we need space to store ancillas. Moreover, our approach of computation with the surface code requires preparation of magic states [41], which requires additional space. In our layout scheme, there are two contributors to the ancilla area:

1. **CNOT ancilla area:** Additional space is needed to store newly initialized ancilla qubits to perform CNOT operations between any two hole pairs.

2. **Operator ancilla area:** Additional space is required to distill and store magic states. Specifically, magic states are needed to perform the S, $S^\dagger$, T, and $T^\dagger$ operations.

We will now determine how many extra hole pairs are required for computation.

11.3.1 Logical CNOT Space

We will assume that all data is represented by a pair of *smooth* holes. To implement an arbitrary CNOT, we require space for 2 additional hole pairs in our layout. This additional space is initially empty and is initialized prior to performing the CNOT operation(s). Figure 30 shows an illustration of the qubit layout. Dashed lines encircle pairs of holes representing one logical qubit in the memory.

CNOT operations can be done easily between smooth and rough hole pairs by using a braiding procedure in which one hole in a hole pair is grown and shrunk to "move" around another hole in another hole pair. Braiding is illustrated in Figure 31 a) and b). Performing a smooth-smooth CNOT is more complicated, and we will need our ancilla space for smooth-rough qubit conversion.

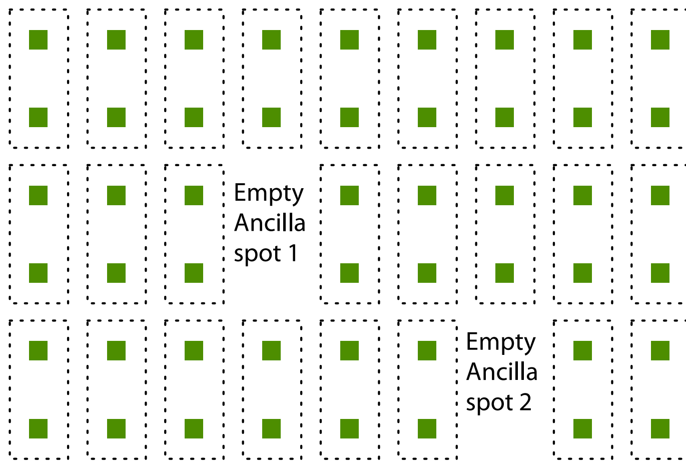
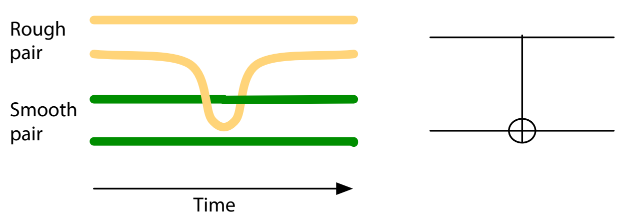
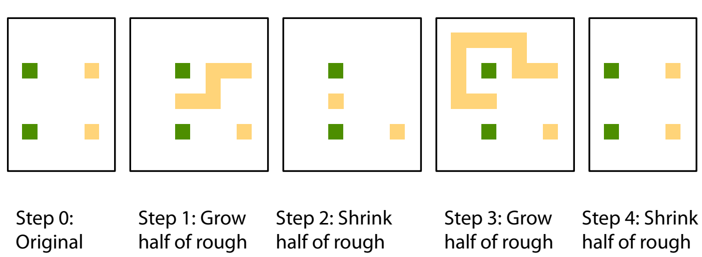


Fig. 30. Logical qubit layout with two empty spots for smooth-smooth CNOT ancilla.



a) Smooth-rough CNOT



b) Smooth-rough CNOT done by braiding

Fig. 31. Smooth-rough CNOT.

The schematic circuit for implementing a smooth-smooth CNOT is shown in Figure 32 a). One of the ancillas is created by puncturing a pair of smooth holes and initializing the qubit in the $|0\rangle$ state. The other ancilla is initialized as a pair of rough holes in the $|+\rangle$ state. This is followed by three smooth-rough CNOTs between the data qubits and ancillas, implemented by braiding, and finally two measurements. By this measurement, the control and one of the ancilla qubits are destroyed and the location of the control qubit moves to one of the ancilla qubits. Figure 32 b) shows how the position of the qubits changes in the layout. This requires us to keep track of the location of the logical qubits throughout the computation. At the end, we are still left with two empty spots, which allows us to perform the next CNOT operation between any two hole pairs. Note that if we add multiple target qubits and replace the middle CNOT in the circuit with a multi-target smooth-rough CNOT, we can implement a smooth-smooth multi-target CNOT. Such circuits are particularly useful for operations such as magic state distillation, as will be described later.

It is important to note that our current resource estimation methodology assumes that multiple logical CNOT gates can be scheduled in parallel, requiring $Qubits.CNOTAncilla = 2Algorithm.paralCNOT$ ancilla locations. If these CNOT gates are scheduled in parallel, we may not be able to guarantee that we can find non-overlapping braids, and some of the CNOTs must be scheduled in sequence. However, our assumption is justified because all the analyzed algorithms have very small parallelization factors of the CNOT operations at the logical level, and it is likely that one can find a few non-overlapping braids between pairs of qubits dispersed in a large system. The feasibility and optimal strategy for parallel scheduling of a larger number of CNOT operations in the surface code is the subject of our ongoing investigation.

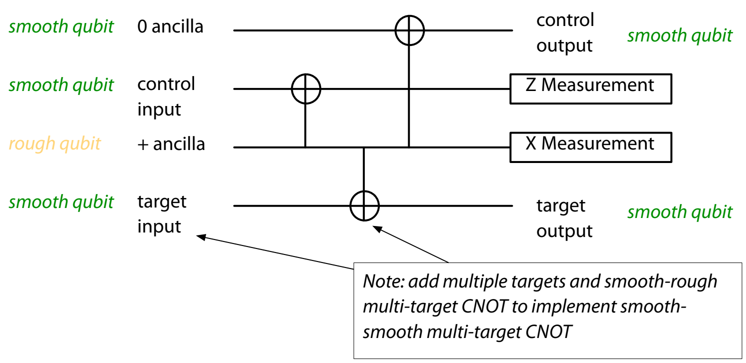
11.3.2 Magic State Distillation Space

Here, we will determine how many ancilla qubits are needed to distill a sufficient number of magic states. The operations that require ancilla qubits are the S, $S^\dagger$, T, and $T^\dagger$ gates. We need two types of ancillas initialized in states $|Y\rangle$ and $|A\rangle$ [37, 41]:

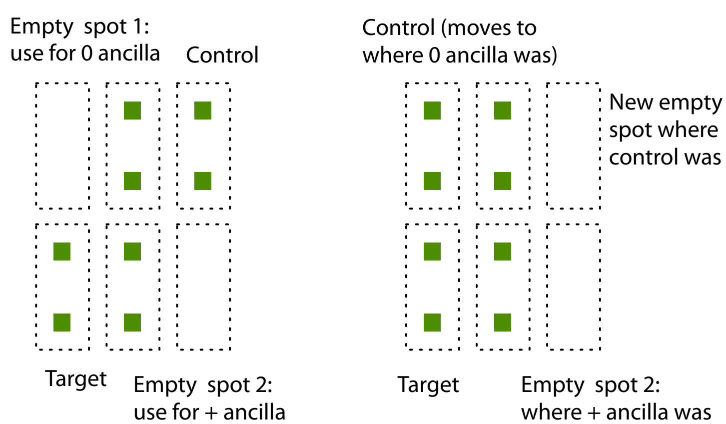
$$|Y\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle + i\,|1\rangle\right), \tag{66}$$

$$|A\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle + e^{i\pi/4}\,|1\rangle\right). \tag{67}$$

Since we can only inject low-precision ancilla states into the system, we have to distill them using distillation circuits that take multiple low-precision ancilla states and output a single ancilla state of a higher precision. The S and $S^\dagger$ gates require one $|Y\rangle$ ancilla and do not destroy the ancilla. The T and $T^\dagger$ gates require one $|A\rangle$ ancilla (which is destroyed in the process of applying the gate) and potentially a $|Y\rangle$ ancilla that is not destroyed as it is used in the application of an S gate. Hence, it is necessary to have enough area to prepare the the $|Y\rangle$ state just a few times (more precisely, $max(Algorithm.paralS, Algorithm.paralT)$ times to allow S and T gate parallelism). The $|Y\rangle$ state preparations are assumed to be at the beginning of the computation. We assume that the $|A\rangle$ ancillas are prepared offline to reduce the amount of time it takes to apply the T and $T^\dagger$ gates. Our methodology makes the simplifying assumption that simultaneous offline preparation of $Algorithm.paralT$ ancillas of



a) Smooth-smooth CNOT using smooth-rough CNOTs



b) Smooth-smooth CNOT moves locations of control and target defect pairs

Fig. 32. Smooth-smooth CNOT.

$|A\rangle$ type is sufficient, but we note that in the real system more ancillas may be need if the T gates are applied frequently.

To compute the resources to prepare the $|Y\rangle$ and $|A\rangle$ states, we need to analyze the distillation circuits for these states. Details on each of the distillation circuits are given below:

- **Distillation of $|Y\rangle$ state:** The distillation circuit for the $|Y\rangle$ takes as input 7 copies of the state $|Y\rangle$ with error probability p and produces a state with an error probability $7p^3$.
- **Distillation of $|A\rangle$ state:** The distillation circuit for the $|A\rangle$ takes as input 15 copies of the state $|A\rangle$ with error probability p and produces a state with an error probability $35p^3$.

Concatenating these circuits allows us to achieve lower error probabilities. To achieve a desired error probability r, we can compute the number of distillation levels L_1 required to distill the $|Y\rangle$ state by solving the equations:

$$\text{YError}(L_1) < r, \tag{68}$$

$$\text{YError}(L_1) = 7\text{YError}(L_1 - 1)^3, \tag{69}$$

where YError(0) is the physical error rate. This yields:

$$L_1 = \left\lceil \frac{log\left(\frac{log(r)+log(\sqrt{7})}{log(YError(0))+3log(\sqrt{7})}\right)}{log(3)} \right\rceil. \tag{70}$$

To solve for the number of distillation levels L_2 required to distill the state $|A\rangle$ we solve the recurrence:

$$\text{AError}(L_2) < r, \tag{71}$$

$$\text{AError}(L_2) = 35\text{AError}(L_2 - 1)^3, \tag{72}$$

where AError(0) is the physical error rate. This yields:

$$L_2 = \left\lceil \frac{log\left(\frac{log(r)+log(\sqrt{35})}{log(AError(0))+3log(\sqrt{35})}\right)}{log(3)} \right\rceil. \tag{73}$$

Once we have computed L_1 and L_2, we can compute the number of extra hole pairs required for the distillation. Let us first consider an abstract distillation circuit where n qubits are distilled into a single qubit. Note that the recursive distillation process can be understood in terms of a tree structure, where the final result is the process of operations involving several layers of source qubits. This is illustrated by example in Figure 33, where $n = 2$ and the qubits involved in $L = 3$ levels of distillation are shown. We can estimate the number of qubits required to facilitate the distillation as n^L, the number of leafs in the tree. Then the number of qubits required to distill a single ancilla $|Y\rangle$ and $|A\rangle$ is 7^{L_1} and

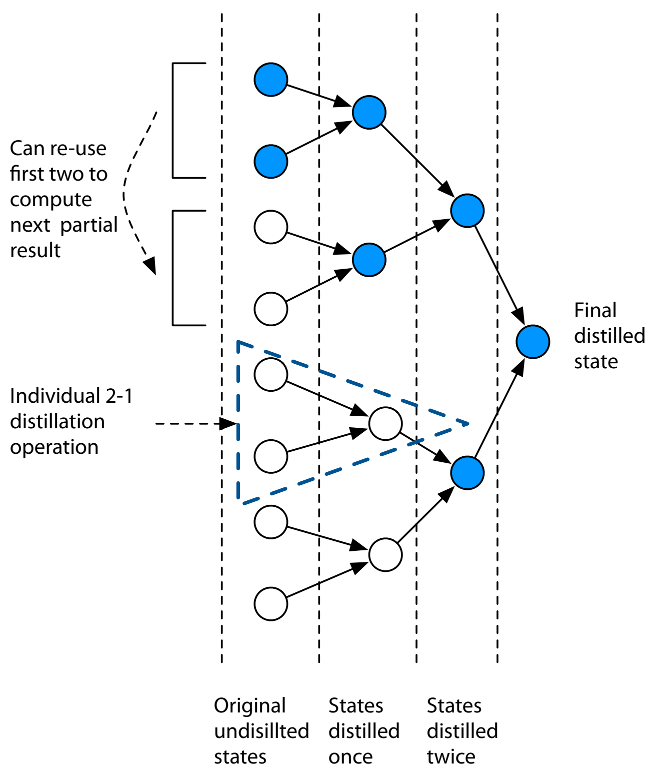


Fig. 33. Tree diagram illustrating state distillation.

15^{L_2} respectively. Finally, let Qubits. |Y⟩ Ancilla and Qubits. |A⟩ Ancilla denote the number of qubits in the entire ancilla factory for ancillas of type $|Y\rangle$ and $|A\rangle$, respectively. Then we have:

$$\text{Qubits.}\,|\text{Y}\rangle\,\text{Ancilla} = 7^{L_1} max(Algorithm.paralS, Algorithm.paralT), \tag{74}$$

$$\text{Qubits.}\,|\text{A}\rangle\,\text{Ancilla} = 15^{L_2} Algorithm.paralT. \tag{75}$$

11.4 Qubit Layout

Here we consider the layout of the physical qubits. It is important to note that our analysis is simplified – we do not consider some connectivity and qubit layout constraints that are specific to each physical technology. We assume that we are able to place qubits in a two dimensional plane on a regular grid, and that nearest neighbors are always able to interact. This assumption greatly simplifies our analysis because we do no have to design a separate qubit layout for each technology. Performing a finer-grained analysis for a specific technology should be a straightforward extension of this work.

We use a square layout, where we pack x hole pair locations into a grid of $\lceil\sqrt{x}\rceil \times \lceil\sqrt{x}\rceil$ hole pair locations with adequate spacing near the edges of the grid to allow braiding operations. Let the width of the grid be denoted as $w = \lceil\sqrt{x}\rceil$. The layout is shown in Figure 34 a). An alternative approach would be a linear layout shown in Figuer 34 b), which may be required by some physical realizations. We do not consider the linear layout in this work as we believe it is too restrictive for any viable quantum architecture.

Each of the blocks shown in Figure 34 a) consists of $d \times d$ physical data qubits to guarantee sufficient code distance. Each block must also contain sufficient number of additional qubits that serve as ancillas during syndrome measurement. Considering only the data qubits, the width and height of the layout are:

$$\text{Layout.width(x)} = d(3\lceil\sqrt{x}\rceil + (\lceil\sqrt{x}\rceil - 1) + 4), \tag{76}$$

$$\text{Layout.height(x)} = d(7\lceil\sqrt{x}\rceil + (\lceil\sqrt{x}\rceil - 1) + 4)). \tag{77}$$

$$\tag{78}$$

This means that the number of unit squares in the surface code layout is $Layout.width \times Layout.height$. The unit squares are the basic building blocks of the code consisting of four data qubits, one on each of the four sides of the square. The only remaining task is to estimate the number of physical qubits that reside in the unit square in the surface code, including the ancillas used for syndrome measurement. This is done separately for the three syndrome extraction methods. We note that some care must be taken to avoid qubit double counting as two squares are incident on each data qubit.

11.4.1 Steane Syndrome Extraction

The circuit for Steane syndrome extraction is illustrated in Figure 35. The circuit on the left measures the site $XXXX$ stabilizer and the circuit on the right measures the plaquette $ZZZZ$ stabilizer. The four measured data qubits are denoted d_1 to d_4, and the ancilla required for the syndrome measurement is denoted a_1. We see that a single ancilla suffices for each site and plaquette stabilizer measurement. Therefore, our proposed qubit layout places one qubit in the center of each site and plaquette. This is shown in Figure 36.

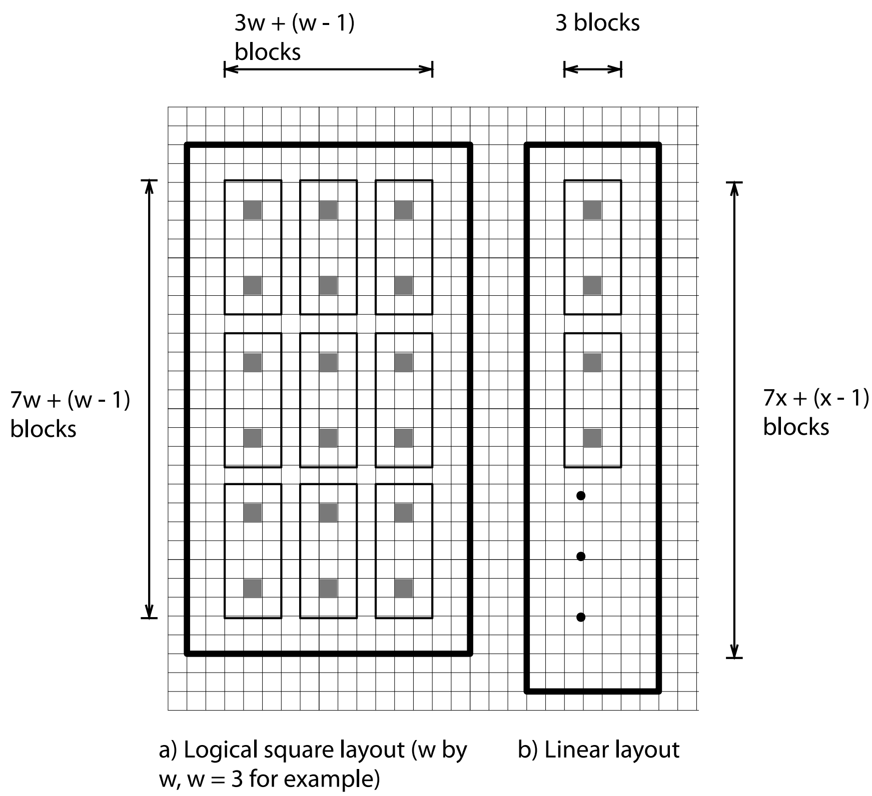


Fig. 34. Square and linear layouts.

In our layout, the number of physical qubits including ancillas in each square is therefore:

$$\text{Layout.QubitsPerSquareSteane} \quad = \quad 4. \tag{79}$$

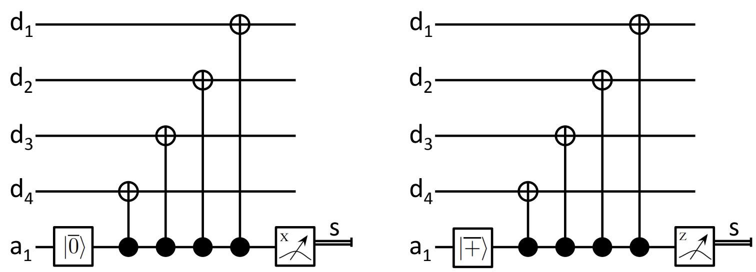


Fig. 35. Steane syndrome extraction circuit.

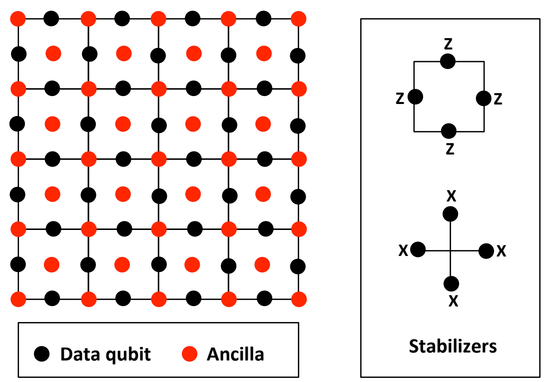


Fig. 36. Qubit and ancilla layout for Steane syndrome extraction.

11.4.2 Shor Syndrome Extraction

The circuit for Shor syndrome extraction is illustrated in Figure 37. The circuit on the left measures the site $XXXX$ stabilizer and the circuit on the right measures the plaquette $ZZZZ$ stabilizer. The four measured qubits are denoted d_1 to d_4, Unlike for the Steane method, a single syndrome measurements requires five ancilla qubits. Four qubits denoted a_1 to a_4 are used to prepare an entangled cat state and the fifth ancillary qubit a_5 is used to verify the state. Our proposed qubit layout places five qubits around the center of each site and plaquette. This is shown in Figure 38.

In our layout, the number of physical qubits including ancillas in each square is therefore:

$$\text{Layout.QubitsPerSquareShor} \quad = \quad 12. \tag{80}$$

11.4.3 Knill Syndrome Extraction

The circuit for Knill syndrome extraction is illustrated in Figure 39. One type of circuit is needed to measure each data qubit in the lattcie. The original data qubit denoted d_1 is

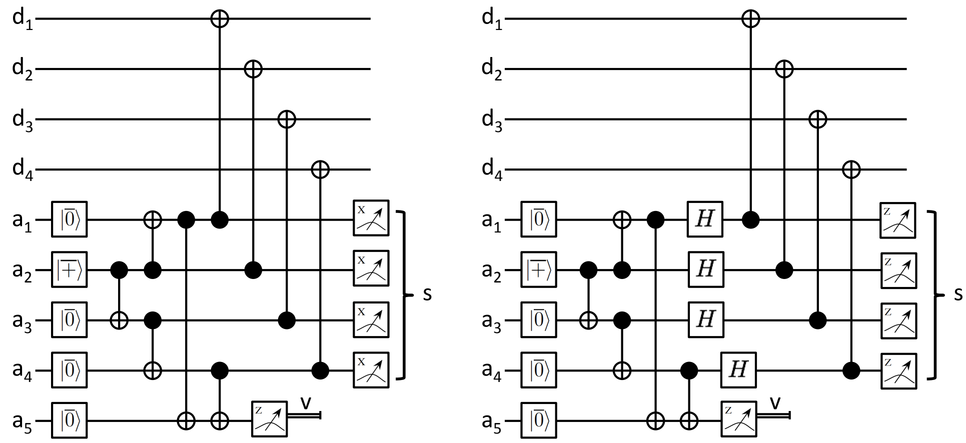


Fig. 37. Shor syndrome extraction circuit.

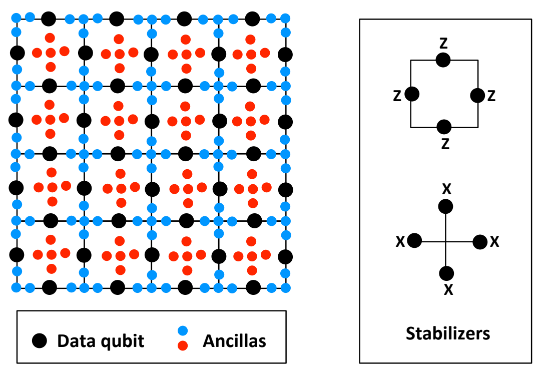


Fig. 38. Qubit and ancilla layout for Shor syndrome extraction.

teleported after measurement and replaces the ancillary qubit a_2. The two other qubits are measured in the process. Our proposed qubit layout places two ancillary qubits right next to each data qubit. This is shown in Figure 40. A black qubit is a data qubit, and the red and blue qubit directly to the right are its ancillas. After measurement, the state of the qubit is teleported to the red qubit. In the next measurement round, the two qubits that were just measured (d_1 and a_1) become the ancillas and the state is teleported back to the original location (into qubit d_1). Note that a further optimization of this layout is possible. For quantum technologies with time consuming state initialization, additional qubits can be used to eliminate the need to wait for ancilla initialization. However, this optimization is beyond the scope of this report.

In our layout, the number of physical qubits including ancillas in each square is therefore:

$$\text{Layout.QubitsPerSquareKnill} \quad = \quad 6. \tag{81}$$

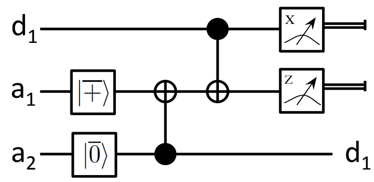


Fig. 39. Knill syndrome extraction circuit.

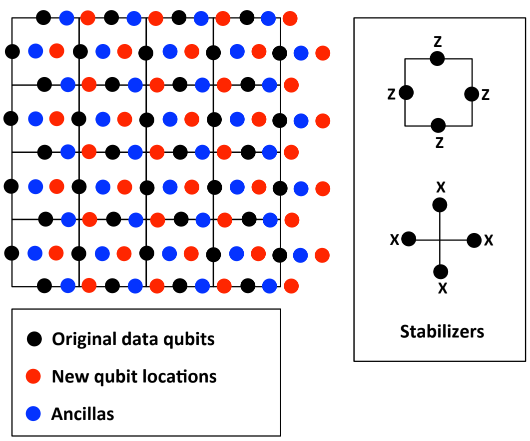


Fig. 40. Qubit and ancilla layout for Knill syndrome extraction.

11.5 Total Area

We can now arrive at the total area required to perform our computation. To recap, we will need space for:

- **Logical qubits:** the algorithm uses Algorithm.numQubits logical qubits.

- **CNOT area:** we need Qubits.CNOTAncilla ancilla spaces for parallel smooth-smooth CNOT operations.

- **Persistent $|Y\rangle$ storage:** One space each to store the $max(Algorithm.paralS, Algorithm.paralT)$ ancillas of type $|Y\rangle$ that are used (and not destroyed) by the S and $T^\dagger$ operators.

- **Ancilla space for $|A\rangle$ distillation:** offline $|A\rangle$ ancilla distillation requires Qubits. |A⟩ Ancilla spaces.

- **Space for the initial $|Y\rangle$ distillation (can overlap with logical qubits and $|A\rangle$ distillation space):** Initially, Qubits. |Y⟩ Ancilla spaces are required to build persistent $|Y\rangle$ state used by the S and $S^\dagger$ operators. However, this space can overlap with the space for the logical qubits and the ancilla space for the $|A\rangle$ state, as the space will be used only once at the beginning of the computation. After this, the space can be used for storing other qubits.

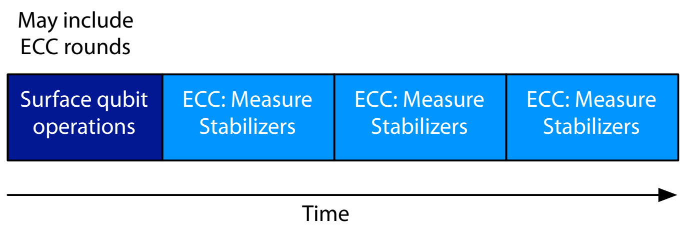


Fig. 41. Surface code logical operation overview.

Putting these together, we get the following expression for the number of spaces required to implement the algorithm:

$$\begin{aligned}\text{Hole Pairs} \quad = \quad & \text{Qubits.CNOTAncilla} + \max(\text{Algorithm.paralS}, \text{Algorithm.paralT}) + \\ & \max\{\text{Qubits.}\,|\text{A}\rangle\,\text{Ancilla} + \text{Algorithm.numQubits}, \text{Qubits.}\,|\text{Y}\rangle\,\text{Ancilla}\}\,.\end{aligned} \tag{82}$$

Finally, we can get the total number of qubits for the Steane extraction technique as:

$$\begin{aligned}\text{Physical Qubits} \quad = \quad & \text{Layout.QubitsPerSquareSteane} \times \text{Layout.width(Hole Pairs)} \times \\ & \text{Layout.height(Hole Pairs)}.\end{aligned} \tag{83}$$

The physical qubit count for Shor and Knill techniques is obtained analogously.

12 Running Time Estimate

Surface code computation requires us to perform logical operations on our surface of qubits and also regular error-correction cycles. To compute the time estimate for the algorithm, we first determine the execution times for the key gates in our circuit.

We build our estimates of the time required to implement logical operations from the ground up. First, we arrive at estimates of simple logical operations. Then, more complex logical operations are expressed in terms of the simpler operations. In the surface code, operations usually consist of some set of physical operations followed by d rounds of error correction to maintain fault tolerance, as shown in Figure 41. Our models quantify the cost of both the logical operations and the error correction.

In the interest of readability, we will assume that our variables refer to times in this section and we will abbreviate the quantum technology parameters as follows:

$$\text{Prep}\,|0\rangle = \text{Technology.Prep}\,|0\rangle\ \text{Time} \tag{84}$$
$$\text{Prep}\,|+\rangle = \text{Technology.Prep}\,|+\rangle\ \text{Time} \tag{85}$$
$$\text{H} = \text{Technology.HTime} \tag{86}$$
$$\text{S} = \text{Technology.STime} \tag{87}$$
$$\text{S}^\dagger = \text{Technology.S}^\dagger\text{Time} \tag{88}$$
$$\text{T} = \text{Technology.TTime} \tag{89}$$
$$\text{T}^\dagger = \text{Technology.T}^\dagger\text{Time} \tag{90}$$
$$\text{CNOT} = \text{Technology.CNOTTime} \tag{91}$$
$$\text{R}_\text{z} = \text{Technology.R}_\text{z}\text{Time} \tag{92}$$

The following list discusses the estimates for a number of logical operations. Since we use smooth hole pairs to encode our logical qubits, we will successively build up to estimates for key "double smooth" logical operations. After this, we can write an equation for the runtime of the algorithm.

1. **No-op (stabilizer measurement cycle):** All the stabilizers (site and plaquette) are measured. By measuring in the right order, these can be parallelized. The time is given by the time of the stabilizer measurement circuit with the maximum time. Since the syndrome measurement circuit is different for each of the three syndrome extraction methods (Steane, Shor and Knill), we have three different expressions for the time to perform the measurements.

$$\text{Steane.EC} = max(\text{Prep}\,|0\rangle + \text{MeasX}, \text{Prep}\,|+\rangle + \text{MeasZ}) + 4\text{CNOT} \tag{93}$$
$$\text{Shor.EC} = max(\text{Prep}\,|0\rangle, \text{Prep}\,|+\rangle) + 4\text{CNOT} + \text{H} + max(\text{MeasX}, \text{MeasZ}) \tag{94}$$
$$\text{Knill.EC} = max(\text{Prep}\,|0\rangle, \text{Prep}\,|+\rangle) + 2\text{CNOT} + max(\text{MeasX}, \text{MeasZ}) \tag{95}$$
$$\tag{96}$$

   In the interest of readability, we will drop the name of the syndrome measurement technique and simply refer to the time needed to perform syndrome measurement as EC.

2. **Error-correction cycles for fault tolerance:** The surface code requires us to maintain d spacing not just in the spatial domain, but also in the temporal domain. To do this, we can apply a primitive that incorporates d stabilizer measurement cycles called TEC. We will add this to several of our fault-tolerant operations.

$$\text{TEC} = \text{EC} \times d \tag{97}$$

3. **Prepare single logical hole:** While we assume all logical data is to be stored in smooth qubits, it is still necessary to prepare rough qubits to implement CNOT operations. We assume that state preparations are followed by the error correction block.

- **Prepare smooth $|0_L\rangle$ and $|+_L\rangle$:** The former requires a set of parallel X measurements in the hole and a round of error correction with new stabilizers for sides of the hole. For the latter, the procedure is similar but instead of X measurements we must prepare the physical $|+\rangle$ state in the block. Since error correction is incorporated, this part is bounded by amount of error correction required for fault tolerance.

$$\mathrm{PrepSmooth}\,|0_\mathrm{L}\rangle \quad = \quad \mathrm{MeasX} + \mathrm{TEC} \tag{98}$$
$$\mathrm{PrepSmooth}\,|+_\mathrm{L}\rangle \quad = \quad \mathrm{Prep}\,|+\rangle + \mathrm{TEC} \tag{99}$$

- **Prepare rough $|0_L\rangle$ and $|+_L\rangle$:** The procedures for these are similar to those for the smooth qubits.

$$\mathrm{PrepRough}\,|0_\mathrm{L}\rangle \quad = \quad \mathrm{Prep}\,|0\rangle + \mathrm{TEC} \tag{100}$$
$$\mathrm{PrepRough}\,|+_\mathrm{L}\rangle \quad = \quad \mathrm{MeasZ} + \mathrm{TEC} \tag{101}$$

4. **Measure logical hole in X or Z:** These operations each require a chain of measurements in either the X or Z bases followed by an ECC round. This will destroy the holes. Each of the measurements in the chain can be done in parallel.

   - **Smooth holes:**

$$\mathrm{MeasXSmooth} \quad = \quad \mathrm{MeasX} + \mathrm{TEC} \tag{102}$$
$$\mathrm{MeasZSmooth} \quad = \quad \mathrm{MeasZ} + \mathrm{TEC} \tag{103}$$

   - **Rough holes:**

$$\mathrm{MeasXRough} \quad = \quad \mathrm{MeasZ} + \mathrm{TEC} \tag{104}$$
$$\mathrm{MeasZRough} \quad = \quad \mathrm{MeasX} + \mathrm{TEC} \tag{105}$$

5. **Grow hole:** Growing a hole requires measurements and corrections.

   - **Smooth:** To grow a smooth hole, X measurements are applied in the region adjacent to the hole and then Z flips are applied based upon the measurements. This also requires measuring three term stabilizers on the sides on the hole, however, this is taken into account by an incorporated ECC step.

$$\mathrm{GrowSmooth} \quad = \quad \mathrm{MeasX} + \mathrm{Z} + \mathrm{TEC} \tag{106}$$

   - **Rough:** Rough holes are done similarily, but with Z measurements and X flips.

$$\mathrm{GrowRough} \quad = \quad \mathrm{MeasZ} + \mathrm{X} + \mathrm{TEC} \tag{107}$$

6. **Shrink hole:** Shrinking or splitting a hole requires us to reenforce the stabilizers in the region that are no longer part of the hole. For smooth holes, measuring the Z stabilizers and possibly correcting them by bit flips on the qubits, followed by error correction

completes the shrinking operation. Rough holes are shrunk analogously. finishes the movement.

$$\begin{aligned} \text{ShrinkSmooth} &= \text{MeasZ} + \text{X} + \text{TEC} && (108) \\ \text{ShrinkRough} &= \text{MeasX} + \text{Z} + \text{TEC} && (109) \end{aligned}$$

7. **Double holes**: It is useful to use double holes to encode information. In general, the costs of building double holes are equivalent to building single holes, as the operations involved in this process can be done in parallel. The costs for several double hole operations are shown below.

$$\begin{aligned} \text{PrepDoubleSmooth}\,|0_\text{L}\rangle &= \text{PrepSmooth}\,|0_\text{L}\rangle && (110) \\ \text{PrepDoubleSmooth}\,|+_\text{L}\rangle &= \text{PrepSmooth}\,|+_\text{L}\rangle && (111) \\ \text{MeasXDoubleSmooth} &= \text{MeasXSmooth} && (112) \\ \text{MeasZDoubleSmooth} &= \text{MeasZSmooth} && (113) \\ \text{PrepDoubleRough}\,|0_\text{L}\rangle &= \text{PrepRough}\,|0_\text{L}\rangle && (114) \\ \text{PrepDoubleRough}\,|+_\text{L}\rangle &= \text{PrepRough}\,|+_\text{L}\rangle && (115) \\ \text{MeasZDoubleRough} &= \text{MeasZRough} && (116) \\ \text{MeasXDoubleRough} &= \text{MeasXRough} && (117) \end{aligned}$$

8. **Low-level state injection (single smooth hole)**: This procedure (as described in [41]) allows us to inject a small smooth hole with an arbitrary state $|\psi\rangle$ into the surface. It requires one Z measurement, the application of four X operators (which can be done in parallel), the application of the operators that applies $|\psi\rangle$, an X stabilizer measurement, then a Z operator, and then four Z operators (which can be done in parallel). An ECC step is added at the end of this procedure.

$$\text{InjectSmooth}\,|\psi_L\rangle = \text{MeasZ} + \text{X} + \text{Apply}\,|\psi\rangle + \text{EC} + \text{Z} + \text{Z} + \text{TEC} \quad (118)$$

As described earlier, two key physical states that we need to perform the S and T gates are the $|Y\rangle = \frac{1}{\sqrt{2}}(|0\rangle + i\,|1\rangle)$ and $|A\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle + e^{i\pi/4}\,|1\rangle\right)$ states. These are assumed to be done using a arbitrary Z rotation mechanism on the physical. Hence, the costs for injecting these gates are as follows.

$$\begin{aligned} \text{InjectSmooth}\,|\text{A}_\text{L}\rangle &= \text{MeasZ} + \text{X} + \text{R}_\text{z} + \text{EC} + 2\text{Z} + \text{TEC} && (119) \\ \text{InjectSmooth}\,|\text{Y}_\text{L}\rangle &= \text{MeasZ} + \text{X} + \text{R}_\text{z} + \text{EC} + 2\text{Z} + \text{TEC} && (120) \end{aligned}$$

9. **Preparing double smooth $|Y_L\rangle$ and $|A_L\rangle$:** The cost to inject these states is given by the cost to prepare the states and to grow and split them. The states are initial single holes, grown as single holes, and split as single holes, building a double hole. These are shown below.

$$\begin{aligned} \text{PrepDoubleSmooth}\,|\text{A}_\text{L}\rangle &= \text{InjectSmooth}\,|\text{A}_\text{L}\rangle + \text{GrowSmooth} + \text{ShrinkSmooth} && (121) \\ \text{PrepDoubleSmooth}\,|\text{Y}_\text{L}\rangle &= \text{InjectSmooth}\,|\text{Y}_\text{L}\rangle + \text{GrowSmooth} + \text{ShrinkSmooth} && (122) \end{aligned}$$

10. **Double smooth-rough CNOT:** A smooth-rough CNOT between two hole pairs is done by braiding. The braiding operation requires two movements and is done using one of the rough hole pieces. Hence, the operation requires a grow, a shrink, another grow, and another shrink.

$$\begin{aligned}\text{DoubleSmoothRoughCNOT} &= \text{GrowRough} + \text{ShrinkRough} + \\ &\quad \text{GrowRough} + \text{ShrinkRough}\end{aligned} \tag{123}$$

11. **Double smooth-rough multi-target CNOT:** In our layout, a multi-target smooth rough CNOT operation involving n targets can be done with n moves to each target pair, followed by one movement back. In theory, this can be done with fewer moves, but this requires the hole pairs to be laid out in manner so segment of the braiding path intersects with itself.

$$\text{DoubleSmoothRoughMTCNOT}(n) = (n+1)(\text{GrowRough} + \text{ShrinkRough}) \tag{124}$$

12. **Double smooth-smooth CNOT:** A smooth-smooth CNOT between hole pairs is done using the construction from [37], which requires a rough ancilla state and a smooth ancilla state. The rough ancilla pair is prepared to a $|+_L\rangle$, the smooth pair is prepared to a $|0_L\rangle$. We assume that the ancilla preparation is done offline and doesn't affect the running time. Then, 3 smooth-rough CNOTs are done, and then a smooth and rough measurements are done.

$$\begin{aligned}\text{DoubleSmoothSmoothCNOT} &= 3\text{DoubleSmoothRoughCNOT} + \\ &\quad max(\text{MeasZDoubleSmooth}, \\ &\quad \text{MeasXDoubleRough})\end{aligned} \tag{125}$$

13. **Double smooth-smooth multi-target CNOT:** The difference between this operation and the single-target double smoooth-smooth CNOT is that the smooth-rough CNOT in the middle of the circuit is replaced by a multi-target smooth-rough CNOT operation. Hence, the cost of this operation is given as follows.

$$\begin{aligned}\text{DoubleSmoothSmoothMTCNOT}(n) &= 2\text{DoubleSmoothRoughCNOT} + \\ &\quad \text{DoubleSmoothRoughMTCNOT}(n) + \\ &\quad max(\text{MeasZDoubleSmooth}, \\ &\quad \text{MeasXDoubleRough})\end{aligned} \tag{126}$$

14. **Prepare logical double smooth $|Y_L\rangle$ and $|A_L\rangle$ to a level:** Concatenated distillation of the states $|Y_L\rangle$ and $|A_L\rangle$ stored in smooth holes is necessary to apply the S and T gates. Here we show the costs of these circuits implemented using regular smooth-smooth CNOT operations and also multi-target CNOTs.

    - **Double smooth $|Y_L\rangle$:** The distillation circuit for the $|Y_L\rangle$ state resembles a Steane-code logical encoder with operations ordered backwards, as shown in Figure

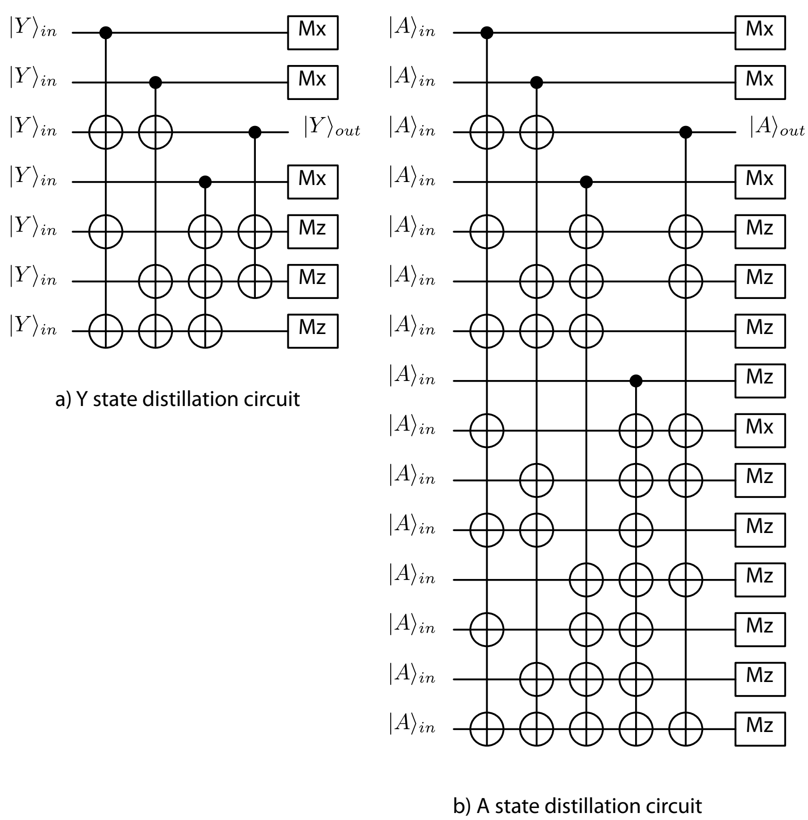


Fig. 42. Distillation circuits.

42 a). Since the probability of needing to stop distillation is asymptotically zero, we will assume that the distillation procedure finishes every time. The cost of a concatenated distillation circuit of a level L is given by the following equations.

$$\begin{aligned}\mathrm{DistillDoubleSmooth}(\left|\mathrm{Y_L}\right\rangle, \mathrm{L}) &= \mathrm{DistillDoubleSmooth}(\left|\mathrm{Y_L}\right\rangle, \mathrm{L}-1) + \\ &\quad 3\mathrm{DoubleSmoothSmoothMTCNOT}(3) + \\ &\quad \mathrm{DoubleSmoothSmoothMTCNOT}(2) + \\ &\quad max(\mathrm{MeasZDoubleSmooth}, \\ &\quad \mathrm{MeasXDoubleSmooth}) \end{aligned} \tag{127}$$

$$\begin{aligned}\mathrm{DistillDoubleSmooth}(\left|\mathrm{Y_L}\right\rangle, 0) &= \mathrm{PrepDoubleSmooth}\left|\mathrm{Y_L}\right\rangle + \\ &\quad 3\mathrm{DoubleSmoothSmoothMTCNOT}(3) + \\ &\quad \mathrm{DoubleSmoothSmoothMTCNOT}(2) + \\ &\quad max(\mathrm{MeasZDoubleSmooth}, \\ &\quad \mathrm{MeasXDoubleSmooth}). \end{aligned} \tag{128}$$

- **Logical $|A\rangle$:** The distillation circuit for the $|A\rangle$ state is shown in Figure 42 b). The cost of a concatenated distillation circuit is given as follows.

$$\begin{aligned}\mathrm{DistillDoubleSmooth}(\left|\mathrm{A_L}\right\rangle, \mathrm{L}) &= \mathrm{DistillDoubleSmooth}(\left|\mathrm{A_L}\right\rangle, \mathrm{L}-1) + \\ &\quad 4\mathrm{DoubleSmoothSmoothMTCNOT}(7) + \\ &\quad \mathrm{DoubleSmoothSmoothMTCNOT}(6) + \\ &\quad max(\mathrm{MeasZDoubleSmooth}, \\ &\quad \mathrm{MeasXDoubleSmooth}) \end{aligned} \tag{129}$$

$$\begin{aligned}\mathrm{DistillDoubleSmooth}\left|\mathrm{A_L}\right\rangle, 0) &= \mathrm{PrepDoubleSmooth}\left|\mathrm{A_L}\right\rangle + \\ &\quad 4\mathrm{DoubleSmoothSmoothMTCNOT}(7) + \\ &\quad \mathrm{DoubleSmoothSmoothMTCNOT}(6) + \\ &\quad max(\mathrm{MeasZDoubleSmooth}, \\ &\quad \mathrm{MeasXDoubleSmooth}) \end{aligned} \tag{130}$$

15. **Double smooth S and $S^\dagger$:** The S gate on a smooth hole is done using the circuit shown in Figure 43 b), which is from [37]. This requires a double smooth ancilla state $|Y_L\rangle$ (which we assumed was prepared at the start of the computation) and the application two Hadamards and two CNOTs. The inverse $S^\dagger$ is obtained by running the circuit backwards.

$$\begin{aligned}\mathrm{DoubleSmoothS} &= 2\mathrm{DoubleSmoothSmoothCNOT} + \\ &\quad 2\mathrm{DoubleSmoothH} \end{aligned} \tag{131}$$

$$\mathrm{DoubleSmoothS}^\dagger = \mathrm{DoubleSmoothS} \tag{132}$$

16. **Double smooth T and $T^\dagger$:** The T gate on a double smooth hole can be done using a measurement-based circuit. This circuit is shown in Figure 43 a). To implement the

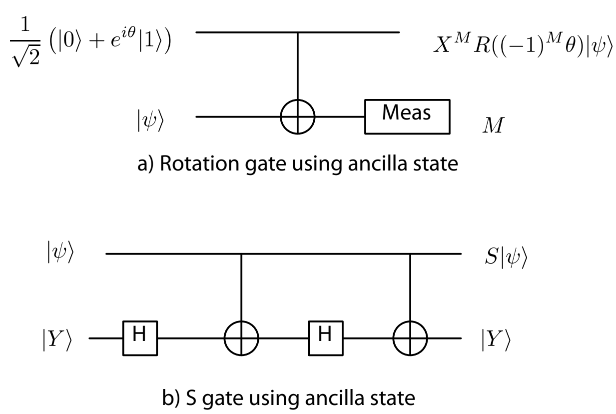


Fig. 43. Measurement-based rotation gate and S gate.

T gate, the input ancilla state $\frac{1}{\sqrt{2}}\left(|0\rangle + e^{i\theta}|1\rangle\right)$ is initialized to be the $|A\rangle$ state. We assume that the ancilla is prepared offline, and it doesn't affect the running time. If the measurement (in the Z basis) indicates that the T gate has been applied, the procedure is done. If not, the $T^\dagger$ operation actually has been applied and an S gate needs to be applied to correct the state. Both events can occur with equal probability. The cost of these operations is given below, the factor 1/2 represents the probability of the application of the S gate.

$$\begin{aligned}\text{DoubleSmoothT} &= \text{DoubleSmoothSmoothCNOT} + \\ &\quad \text{MeasZDoubleSmooth} + \\ &\quad \frac{1}{2}\text{DoubleSmoothS} \qquad (133)\\ \text{DoubleSmoothT}^\dagger &= \text{DoubleSmoothT} \qquad (134)\end{aligned}$$

17. **Double smooth H:** To perform the logical H gate, we can use a measurement-based rotation procedure similar to the procedure for implementing the T gate. However, as described in [37], it is possible to do this with lower overhead using the following procedure. First a patch of the surface surrounding smooth holes is cut out using a chain of Z measurements, followed by an error correction round needed to correct the sign of newly created weight three Z stabilizers. Next, a transversal round of H operations is done inside the patch. A set of physical swap gates are then required to reconnect the patch to the surface by shifting the patch (this requires as many cycles as the size of the patch, which is assumed to be 3d). Then another error correction round needs to be done. At this point, the qubit was converted to a rough one and neets to be converted back. This requires a smooth ancilla in the $|+\rangle$ state, a CNOT gate, followed by a measurement of the rough qubit in the Z basis. The cost of this operation

is detailed below.

$$\begin{aligned}\text{DoubleSmoothH} &= \text{MeasZ} + \text{TEC} + \text{H} + 3d \times 3 \times \text{CNOT} + \text{TEC} + \\ &\quad \text{DoubleSmoothPrep}\,|+\rangle + \text{DoubleRoughSmoothCNOT} + \\ &\quad \text{MeasZDoubleSmooth}\end{aligned} \tag{135}$$

From these, we can determine the execution time of the algorithm as follows.

$$\text{Execution Time} = \text{DistillDoubleSmooth}(|\text{Y}_\text{L}\rangle\,, \text{L}_1) + \tag{136}$$
$$(1/\text{Algorithm.paralPrep}\,|0\rangle) \times \text{Algorithm.numPrep}\,|0\rangle \times \text{PrepDoubleSmooth}\,|0_\text{L}\rangle + \tag{137}$$
$$(1/\text{Algorithm.paralPrep}\,|+\rangle) \times \text{Algorithm.numPrep}\,|+\rangle \times \text{PrepDoubleSmooth}\,|+_\text{L}\rangle + \tag{138}$$
$$(1/\text{Algorithm.paralH}) \times \text{Algorithm.numH} \times \text{DoubleSmoothH} + \tag{139}$$
$$(1/\text{Algorithm.paralS}) \times \text{Algorithm.numS} \times \text{DoubleSmoothS} + \tag{140}$$
$$(1/\text{Algorithm.paralS}^\dagger) \times \text{Algorithm.numS}^\dagger \times \text{DoubleSmoothS}^\dagger + \tag{141}$$
$$(1/\text{Algorithm.paralT}) \times \text{Algorithm.numT} \times \text{DoubleSmoothT} + \tag{142}$$
$$(1/\text{Algorithm.paralT}^\dagger) \times \text{Algorithm.numT}^\dagger \times \text{DoubleSmoothT}^\dagger + \tag{143}$$
$$(1/\text{Algorithm.paralCNOT}) \times \text{Algorithm.numCNOT} \times \text{DoubleSmoothSmoothCNOT} + \tag{144}$$
$$(1/\text{Algorithm.paralMeasX}) \times \text{Algorithm.MeasX} \times \text{MeasXSmooth} + \tag{145}$$
$$(1/\text{Algorithm.paralMeasZ}) \times \text{Algorithm.MeasZ} \times \text{MeasZSmooth} \tag{146}$$

13 Number of Gates

Our remaining task is to estimate the number of gates needed by the entire algorithm. We obtain an accurate gate count for each gate type as follows. First, we estimate the number of gates needed to perform error correction. Since error correction is performed continuously, and all other operations require a small number of gates, this is the dominant factor in the final gate count. The gate count for error correction is estimated for each of the three syndrome extraction techniques (Shor, Steane, Knill) separately. Second, we add the small number of gates required to perform logical operations. Note that these gate counts exclude syndrome measurements (to prevent double counting), and therefore the estimate is independent of the syndrome extraction method of choice.

In the interest of readability, we will assume that our variables refer to gates (i.e., the variable is a vector with each entry corresponding to one gate type). For example, Prep $|0\rangle$ refers to a vector that is almost everywhere zero except a one in the entry that represents the $|0\rangle$ state preparation. Similarly, H represents the Hadamard gate in the above vector notation, etc.

1. **No-op (stabilizer measurement cycle):** All the stabilizers (site and plaquette) are measured. Since the syndrome measurement circuit is different for each of the three syndrome extraction methods (Steane, Shor and Knill), we have three different

expressions for the gate count:

$$\begin{aligned}
\text{Steane.EC} &= \text{Prep}\,|0\rangle + \text{Prep}\,|+\rangle + 8\text{CNOT} + \text{MeasX} + \text{MeasZ} && (147)\\
\text{Shor.EC} &= 8\text{Prep}\,|0\rangle + 2\text{Prep}\,|+\rangle + 18\text{CNOT} + 4\text{MeasX} + 6\text{MeasZ} + 4\text{H} && (148)\\
\text{Knill.EC} &= \text{Prep}\,|0\rangle + \text{Prep}\,|+\rangle + 2\text{CNOT} + \text{MeasX} + \text{MeasZ} && (149)
\end{aligned}$$

Below we express the number of gates that need to be performed (excluding syndrome measurement) to implement each gate. We follow the same notation as in the previous section.

2. **Error-correction cycles for fault tolerance**

$$\text{TEC} = \text{EC} \times d \qquad (150)$$

3. **Prepare single logical hole:**

   - **Prepare smooth $|0_L\rangle$ and $|+_L\rangle$**

$$\begin{aligned}
\text{PrepSmooth}\,|0_\text{L}\rangle &= d \times d \times \text{MeasX} && (151)\\
\text{PrepSmooth}\,|+_\text{L}\rangle &= d \times d \times \text{Prep}\,|+\rangle && (152)
\end{aligned}$$

   - **Prepare rough $|0_L\rangle$ and $|+_L\rangle$**

$$\begin{aligned}
\text{PrepRough}\,|0_\text{L}\rangle &= d \times d \times \text{Prep}\,|0\rangle && (153)\\
\text{PrepRough}\,|+_\text{L}\rangle &= d \times d \times \text{MeasZ} && (154)
\end{aligned}$$

4. **Measure logical hole in X or Z:**

   - **Smooth holes**

$$\begin{aligned}
\text{MeasXSmooth} &= 3 \times d \times d \times \text{MeasX} && (155)\\
\text{MeasZSmooth} &= 8 \times d \times d \times \text{MeasZ} && (156)
\end{aligned}$$

   - **Rough holes**

$$\begin{aligned}
\text{MeasXRough} &= 8 \times d \times d \times \text{MeasZ} && (157)\\
\text{MeasZRough} &= 3 \times d \times d \times \text{MeasX} && (158)
\end{aligned}$$

5. **Grow hole** It is not possible to accurately quantify the number of gates when the size of the region into which the hole grows is unknown. However, the number of gates does not contribute significantly to the final gate count, and is reported here as 0.

$$\text{GrowSmooth} = 0 \qquad (159)$$

$$\text{GrowRough} = 0 \qquad (160)$$

6. **Shrink hole** Similarly as for hole growth we can assume that the additional gate count is negligible.

$$\begin{aligned} \text{ShrinkRough} &= 0 && (161) \\ \text{ShrinkSmooth} &= 0 && (162) \end{aligned}$$

7. **Double holes**:

$$\begin{aligned} \text{PrepDoubleSmooth}\,|0_\text{L}\rangle &= 2\text{PrepSmooth}\,|0_\text{L}\rangle && (163) \\ \text{PrepDoubleSmooth}\,|+_\text{L}\rangle &= 2\text{PrepSmooth}\,|+_\text{L}\rangle && (164) \\ \text{MeasXDoubleSmooth} &= \text{MeasXSmooth} && (165) \\ \text{MeasZDoubleSmooth} &= 2\text{MeasZSmooth} && (166) \\ \text{PrepDoubleRough}\,|0_\text{L}\rangle &= 2\text{PrepRough}\,|0_\text{L}\rangle && (167) \\ \text{PrepDoubleRough}\,|+_\text{L}\rangle &= 2\text{PrepRough}\,|+_\text{L}\rangle && (168) \\ \text{MeasZDoubleRough} &= \text{MeasZRough} && (169) \\ \text{MeasXDoubleRough} &= 2\text{MeasXRough} && (170) \end{aligned}$$

8. **Low-level state injection (single smooth hole)**:

$$\text{InjectSmooth}\,|\text{A}_\text{L}\rangle = \text{MeasZ} + 4\text{X} + 6\text{Z} \qquad (171)$$
$$\text{InjectSmooth}\,|\text{Y}_\text{L}\rangle = \text{MeasZ} + 4\text{X} + 6\text{Z} \qquad (172)$$

9. **Preparing double smooth $|Y_L\rangle$ and $|A_L\rangle$**

$$\text{PrepDoubleSmooth}\,|\text{A}_\text{L}\rangle = \text{InjectSmooth}\,|\text{A}_\text{L}\rangle \qquad (173)$$
$$\text{PrepDoubleSmooth}\,|\text{Y}_\text{L}\rangle = \text{InjectSmooth}\,|\text{Y}_\text{L}\rangle \qquad (174)$$

10. **Double smooth-rough CNOT**

$$\text{DoubleSmoothRoughCNOT} = 0 \qquad (175)$$
$$(176)$$

11. **Double smooth-rough multi-target CNOT**

$$\text{DoubleSmoothRoughMTCNOT}(n) = 0 \qquad (177)$$

12. **Double smooth-smooth CNOT**

$$\begin{aligned} \text{DoubleSmoothSmoothCNOT} = {} & \text{PrepDoubleSmooth}\,|0_\text{L}\rangle + && (178) \\ & \text{PrepDoubleRough}\,|+_\text{L}\rangle + \\ & \text{MeasZDoubleSmooth} + \\ & \text{MeasXDoubleRough} \end{aligned}$$

13. **Double smooth-smooth multi-target CNOT**

$$\begin{aligned}\text{DoubleSmoothSmoothMTCNOT}(n) &= \text{PrepDoubleSmooth}\left|0_{\text{L}}\right\rangle + \\ &\quad \text{PrepDoubleRough}\left|+_{\text{L}}\right\rangle + \\ &\quad \text{MeasZDoubleSmooth} + \\ &\quad \text{MeasXDoubleRough}\end{aligned} \tag{179}$$

14. **Prepare logical double smooth $|Y_L\rangle$ and $|A_L\rangle$ to a level.**

- **Double smooth $|Y_L\rangle$**

$$\begin{aligned}\text{DistillDoubleSmooth}(\left|\text{Y}_{\text{L}}\right\rangle, \text{L}) &= 7\text{DistillDoubleSmooth}(\left|\text{Y}_{\text{L}}\right\rangle, \text{L}-1) + \\ &\quad 3\text{DoubleSmoothSmoothMTCNOT}(3) + \\ &\quad \text{DoubleSmoothSmoothMTCNOT}(2) + \\ &\quad 3\text{MeasZDoubleSmooth} + \\ &\quad 3\text{MeasXDoubleSmooth}\end{aligned} \tag{180}$$

$$\begin{aligned}\text{DistillDoubleSmooth}(\left|\text{Y}_{\text{L}}\right\rangle, 0) &= 7\text{PrepDoubleSmooth}\left|\text{Y}_{\text{L}}\right\rangle + \\ &\quad 3\text{DoubleSmoothSmoothMTCNOT}(3) + \\ &\quad \text{DoubleSmoothSmoothMTCNOT}(2) + \\ &\quad 3\text{MeasZDoubleSmooth} + \\ &\quad 3\text{MeasZDoubleSmooth}.\end{aligned} \tag{181}$$

- **Logical $|A\rangle$**

$$\begin{aligned}\text{DistillDoubleSmooth}(\left|\text{A}_{\text{L}}\right\rangle, \text{L}) &= 15\text{DistillDoubleSmooth}(\left|\text{A}_{\text{L}}\right\rangle, \text{L}-1) + \\ &\quad 5\text{DoubleSmoothSmoothMTCNOT}(7) + \\ &\quad \text{DoubleSmoothSmoothMTCNOT}(6) + \\ &\quad 10\text{MeasZDoubleSmooth} + \\ &\quad 4\text{MeasXDoubleSmooth}\end{aligned} \tag{182}$$

$$\begin{aligned}\text{DistillDoubleSmooth}\left|\text{A}_{\text{L}}\right\rangle, 0) &= 15\text{PrepDoubleSmooth}\left|\text{A}_{\text{L}}\right\rangle + \\ &\quad 5\text{DoubleSmoothSmoothMTCNOT}(7) + \\ &\quad \text{DoubleSmoothSmoothMTCNOT}(6) + \\ &\quad 10\text{MeasZDoubleSmooth} + \\ &\quad 4\text{MeasXDoubleSmooth}.\end{aligned} \tag{183}$$

15. **Double smooth S and $S^\dagger$**

$$\begin{aligned}\text{DoubleSmoothS} &= 2\text{DoubleSmoothSmoothCNOT} + \\ &\quad 2\text{DoubleSmoothH}\end{aligned} \tag{184}$$

$$\text{DoubleSmoothS}^\dagger = \text{DoubleSmoothS} \tag{185}$$

16. **Double smooth T and $T^\dagger$**

$$\begin{aligned} \text{DoubleSmoothT} &= \text{DistillDoubleSmooth}(|\text{A}_\text{L}\rangle, \text{L}_2) + \\ &\quad \text{DoubleSmoothSmoothCNOT} + \\ &\quad \text{MeasZDoubleSmooth} + \\ &\quad \frac{1}{2}\text{DoubleSmoothS} \end{aligned} \tag{186}$$

$$\text{DoubleSmoothT}^\dagger = \text{DoubleSmoothT}. \tag{187}$$

17. **Double smooth H**

$$\begin{aligned} \text{DoubleSmoothH} &= 20 \times d \times \text{MeasZ} + 21 \times d \times d \times \text{H} + 3d \times 3 \times \text{CNOT} + \\ &\quad \text{DoubleSmoothPrep}\,|+\rangle + \text{DoubleRoughSmoothCNOT} + \\ &\quad \text{MeasZDoubleSmooth}. \end{aligned} \tag{188}$$

By dividing the execution time and time needed to do one error correction obtained in the previous section, we obtain the number of error-correction cycles, denoted EC Cycles. From the above equations, we can determine the total gate count during the execution of the algorithm.

$$\text{Gate Count} = \text{Physical Area} \times \text{EC Cycles} \times \text{EC} + \tag{189}$$
$$\text{DistillDoubleSmooth}(|\text{Y}_\text{L}\rangle, \text{L}_1) + \tag{190}$$
$$\text{Algorithm.numPrep}\,|0\rangle \times \text{PrepDoubleSmooth}\,|0_\text{L}\rangle + \tag{191}$$
$$\text{Algorithm.numPrep}\,|+\rangle \times \text{PrepDoubleSmooth}\,|+_\text{L}\rangle + \tag{192}$$
$$\text{Algorithm.numH} \times \text{DoubleSmoothH} + \tag{193}$$
$$\text{Algorithm.numS} \times \text{DoubleSmoothS} + \tag{194}$$
$$\text{Algorithm.numS}^\dagger \times \text{DoubleSmoothS}^\dagger + \tag{195}$$
$$\text{Algorithm.numT} \times \text{DoubleSmoothT} + \tag{196}$$
$$\text{Algorithm.numT}^\dagger \times \text{DoubleSmoothT}^\dagger + \tag{197}$$
$$\text{Algorithm.numCNOT} \times \text{DoubleSmoothSmoothCNOT} + \tag{198}$$
$$\text{Algorithm.MeasX} \times \text{MeasXSmooth} + \tag{199}$$
$$\text{Algorithm.MeasZ} \times \text{MeasZSmooth} \tag{200}$$

14 Software Tools for Resource Estimation

The QuRE toolbox is implemented as a suite of Octave scripts that automatically generate the resource estimates for a cross product of algorithms, quantum technologies, and error-correction techniques. The software processes the following inputs. For each analyzed algorithm, we require the number of logical gates of each type, the parallelization factor for these gates (how many gates of each type can be safely scheduled in parallel), and the number of logical qubits. These inputs are located in the "Alg_*.m" files. Another required input is information about each combination of physical quantum technology and control protocol. The required information is the time needed by each gate type, the error of the worst gate, and the error rate per unit of time. These inputs are located in a separate "PMD_*.m" file for each quantum technology / control protocol combination.

The heart of the software are the recursive relations specifying the amount of resources needed by the error-correcting codes at each level of concatenation, as well as the equations specifying the behavior of the surface code. These relations appear in this report, and are implemented in the "ECC_*.m" scripts.

Results for all combinations of algorithms, physical quantum technologies, error correction protocols (the Chinese menu) are generated by running the "Print.m" script. Estimates are output in the out directory and the table directory. In the out directory, one file is generated for each entry in the Chinese menu. The table directory includes results in a succinct form – these results are also shown in Section 15 of this report.

A typical output in the "out" directory includes the following entries:

- Information if the error-correction threshold is met, and the target level of concatenation to meet 50% circuit reliability. In case of the surface code, code distance is reported instead of the concatenation level.
- Probability of success of the computation.
- Number of physical qubits needed. In case of Bacon-Shor code and the Knill's post-selection scheme which use ancilla factories we also report the number of qubits needed by the ancilla factory in addition to the total qubit count.
- Running time of the algorithm in ns.
- The length of a time interval after which idle qubits must be error corrected, and the number of error-correction rounds that must be done in total.
- A detailed gate count that includes the number of occurrences of each physical gate construct during the entire computation. We report the following gate constructs: $CNOT$, $SWAP$, H, $|+>$ *prep.*, $|0>$ *prep.*, $Xmeas.$, $Zmeas.$, X, Y, Z, S and T.

For the Bacon Shor code and the Knill's post-selection scheme which require ancilla factories the following information is reported:

- Time needed to produce and inject an ancillary state.
- A detailed gate count specifying the number of gates needed to produce one good ancillary state.
- The number of gates needed to produce all ancillary states required by all logical S and T gates that occur in the algorithm.

Inspecting the results generated by our tool resulted in the following observations:

- Unlike topological error-correcting codes, the concatenated codes do not meet the threshold of most quantum architectures.
- The number of gates and running time for topological and concatenated codes is comparable.

- The Knill's post-selection scheme has higher threshold than the Bacon-Shor and Steane code, and therefore often requires fewer levels of concatenation and the resource requirements are lower.

The outputs of the software were validated by comparing the outputs against the equations present in this report. Since the data input for all algorithms and quantum technology / control protocol combinations have the same format, it was sufficient to validate results for one algorithm and quantum technology / control protocol combination (we chose the ground state estimation algorithm and superconducting qubits with primitive control). We verified results for all four error-correcting codes.

The outputs for all entries of the Chinese menu appear in the "out" and "table" directory, and selected results are also reported in Section 15.

15 Numerical Results

Here we present an overview of the numerical results obtained by the QuRE toolbox. We only report the most basic properties for each error correcting code, namely the duration and number of gates required to execute a single logical gate at a specified concatenation level or code distance. For each combination of error correcting code, algorithm, and physical quantum technology we also report the number of qubits, number of gates and running time required by the quantum computer. More detailed results can be obtained by running the QuRE toolbox.

The gate time for a logical gate at a specified level of concatenation of the concatenated codes is obtained by QuRE by solving the the recursive equations in Sections 6, 7, and 8. Tables 7, 8, and 9 show the results for the Steane code, the Bacon-Shor code, and the Knill scheme respectively. The tables show results for one to five levels of concatenation. The reported gate times are based on the superconducting quantum technology with primitive control. Table 10 summarizes the gate times for the surface code, again using superconductors with primitive control. The columns of the table show gate times for various choices of code distance. Note that the surface code doesn't need the $SWAP$ operation, and we do not need to distinguish a horizontal or vertical $CNOT$ because the cost of both gates is identical. We observe that while the gate time for concatenated codes increases sharply with increasing level of concatenation, the gate time for the surface code increases only moderately with increasing code distance.

The number of gates of each type required to execute the error correction operation ($\mathcal{EC}$) with the concatenated codes is reported in Tables 11, 12, and 13. The tables show results for one to five levels of concatenation. We report the gate count for the error correction operation because this is the most frequently repeated operation during computation, and error correction is performed as part of any other gate. Detailed gate count for any logical operation at any concatenation level can be obtained from the QuRE toolbox, but we do not report these results here as the cost of error correction is the dominant factor.

The resources required by a quantum computer for each combination of algorithm, physical quantum technology, control protocol and error correcting code appears in Table 15. The first five columns identify the combination that is being evaluated, the next three columns show the resources, and the last column shows the level of concatenation or code distance. Due to space constraints, the algorithm, physical quantum technology, error correcting code, and syndrome extraction method are labeled with abbreviated labels. These abbreviations are summarized in Table 14. The reported resources can be interepreted as follows. The number of qubits is the total number of physical qubits needed by the quantum computer. The number of gates is the total number of physical gates (across all types) that need to be executed. Finally, the running time is the total time in nanoseconds needed to finish the computation.

Table 6. Gate time (in ns) for Steane code.

Operation(s)	**m=1**	**m=2**	**m=3**	**m=4**	**m=5**
$\mathcal{EC}$	876	$2.16e+04$	$5.8e+05$	$1.58e+07$	$4.31e+08$
$hCNOT$	$1.02e+03$	$2.97e+04$	$8.16e+05$	$2.23e+07$	$6.09e+08$
$vCNOT$	$1.00e+03$	$2.85e+04$	$7.76e+05$	$2.11e+07$	$5.77e+08$
$hSWAP$	$1.01e+03$	$2.97e+04$	$8.17e+05$	$2.23e+07$	$6.1e+08$
$vSWAP$	978	$2.74e+04$	$7.44e+05$	$2.03e+07$	$5.53e+08$
H	882	$2.24e+04$	$6.02e+05$	$1.64e+07$	$4.47e+08$
$P_{\|+\rangle}$	$1.28e+03$	$3.05e+04$	$8.19e+05$	$2.23e+07$	$6.1e+08$
$P_{\|0\rangle}$	$1.28e+03$	$3.05e+04$	$8.19e+05$	$2.23e+07$	$6.1e+08$
$\mathcal{M}_X$	876	$2.16e+04$	$5.8e+05$	$1.58e+07$	$4.31e+08$
$\mathcal{M}_Z$	876	$2.16e+04$	$5.8e+05$	$1.58e+07$	$4.31e+08$
X	886	$2.25e+04$	$6.02e+05$	$1.64e+07$	$4.47e+08$
Y	886	$2.25e+04$	$6.02e+05$	$1.64e+07$	$4.47e+08$
Z	877	$2.24e+04$	$6.02e+05$	$1.64e+07$	$4.47e+08$
P_{cat}	$1.17e+03$	$3.29e+04$	$9.02e+05$	$2.46e+07$	$6.72e+08$
S	$5.29e+03$	$1.54e+05$	$4.22e+06$	$1.15e+08$	$3.14e+09$
T	$5.29e+03$	$1.54e+05$	$4.22e+06$	$1.15e+08$	$3.14e+09$

Table 7. Gate time (in ns) for Bacon-Shor code.

Operation(s)	**m=1**	**m=2**	**m=3**	**m=4**	**m=5**
$\mathcal{EC}$	326	$4.13e+03$	$6.05e+04$	$9.21e+05$	$1.42e+07$
$hCNOT$	484	$8.31e+03$	$1.31e+05$	$2.04e+06$	$3.15e+07$
$vCNOT$	484	$8.31e+03$	$1.31e+05$	$2.04e+06$	$3.15e+07$
$hSWAP$	462	$7.83e+03$	$1.23e+05$	$1.91e+06$	$2.94e+07$
$vSWAP$	462	$7.83e+03$	$1.23e+05$	$1.91e+06$	$2.94e+07$
H	828	$1.37e+04$	$2.13e+05$	$3.29e+06$	$5.07e+07$
$P_{\|+\rangle}$	760	$9.13e+03$	$1.33e+05$	$2.03e+06$	$3.12e+07$
$P_{\|0\rangle}$	760	$9.13e+03$	$1.33e+05$	$2.03e+06$	$3.12e+07$
$\mathcal{M}_X$	342	$4.48e+03$	$6.49e+04$	$9.86e+05$	$1.52e+07$
$\mathcal{M}_Z$	336	$4.47e+03$	$6.49e+04$	$9.86e+05$	$1.52e+07$
X	336	$4.47e+03$	$6.49e+04$	$9.86e+05$	$1.52e+07$
Y	337	$4.8e+03$	$6.94e+04$	$1.05e+06$	$1.62e+07$
Z	327	$4.46e+03$	$6.49e+04$	$9.86e+05$	$1.52e+07$
P_{cat}	0	0	0	0	0
S	$2.62e+03$	$4.41e+04$	$6.89e+05$	$1.06e+07$	$1.64e+08$
T	$3.44e+03$	$5.68e+04$	$8.85e+05$	$1.37e+07$	$2.11e+08$

Table 8. Gate time (in ns) for Knill/Steane code.

Operation(s)	**m=1**	**m=2**	**m=3**	**m=4**	**m=5**
$\mathcal{EC}$	912	$5.21e+03$	$5.72e+04$	$6.38e+05$	$7.08e+06$
$hCNOT$	$1.05e+03$	$7.73e+03$	$8.43e+04$	$9.32e+05$	$1.04e+07$
$vCNOT$	$1.04e+03$	$7.53e+03$	$8.12e+04$	$8.97e+05$	$9.97e+06$
$hSWAP$	358	$4.12e+03$	$4.46e+04$	$4.94e+05$	$5.49e+06$
$vSWAP$	324	$3.51e+03$	$3.82e+04$	$4.24e+05$	$4.71e+06$
H	918	$5.44e+03$	$5.89e+04$	$6.57e+05$	$7.3e+06$
$P_{\|+\rangle}$	640	$3.94e+03$	$4.43e+04$	$4.93e+05$	$5.48e+06$
$P_{\|0\rangle}$	640	$3.93e+03$	$4.43e+04$	$4.93e+05$	$5.48e+06$
$\mathcal{M}_X$	912	$5.21e+03$	$5.72e+04$	$6.38e+05$	$7.08e+06$
$\mathcal{M}_Z$	912	$5.21e+03$	$5.72e+04$	$6.38e+05$	$7.08e+06$
X	922	$5.45e+03$	$5.89e+04$	$6.57e+05$	$7.3e+06$
Y	922	$5.45e+03$	$5.89e+04$	$6.57e+05$	$7.3e+06$
Z	913	$5.44e+03$	$5.89e+04$	$6.57e+05$	$7.3e+06$
P_{cat}	$1.21e+03$	$7.98e+03$	$9.03e+04$	$1.01e+06$	$1.12e+07$
S	$5.38e+03$	$3.53e+04$	$3.94e+05$	$4.38e+06$	$4.87e+07$
T	$5.38e+03$	$3.53e+04$	$3.94e+05$	$4.39e+06$	$4.87e+07$

Table 9. Gate time (in ns) for surface code.

Operation(s)	**d=3**	**d=7**	**d=21**	**d=51**	**d=101**
$\mathcal{EC}$	166	166	166	166	166
$CNOT$	$6.71e+03$	$1.53e+04$	$4.56e+04$	$1.1e+05$	$2.18e+05$
H	$4.78e+03$	$1.09e+04$	$3.22e+04$	$7.8e+04$	$1.54e+05$
$P_{\|+\rangle}$	598	$1.26e+03$	$3.59e+03$	$8.57e+03$	$1.69e+04$
$P_{\|0\rangle}$	514	$1.18e+03$	$3.5e+03$	$8.48e+03$	$1.68e+04$
$\mathcal{M}_X$	16	16	16	16	16
$\mathcal{M}_Z$	10	10	10	10	10
S	$2.3e+04$	$5.25e+04$	$1.56e+05$	$3.77e+05$	$7.45e+05$
T	$1.87e+04$	$4.27e+04$	$1.27e+05$	$3.07e+05$	$6.07e+05$

Table 10. Gate count per error correction operation with Steane code.

Operation(s)	**m=1**	**m=2**	**m=3**	**m=4**	**m=5**
$hCNOT$	14	$1.85e+03$	$3.27e+05$	$5.84e+07$	$1.04e+10$
$vCNOT$	28	$3.85e+03$	$6.76e+05$	$1.2e+08$	$2.15e+10$
$hSWAP$	18	$5.29e+03$	$9.7e+05$	$1.74e+08$	$3.11e+10$
$vSWAP$	15	$4.84e+03$	$8.58e+05$	$1.52e+08$	$2.71e+10$
H	7	959	$1.69e+05$	$3e+07$	$5.36e+09$
$P_{\|+\rangle}$	8	$1.06e+03$	$1.87e+05$	$3.34e+07$	$5.95e+09$
$P_{\|0\rangle}$	12	$1.58e+03$	$2.81e+05$	$5e+07$	$8.92e+09$
$\mathcal{M}_X$	20	$2.64e+03$	$4.68e+05$	$8.34e+07$	$1.49e+10$

$\mathcal{M}_Z$	0	0	0	0	0
X	0	0	0	0	0
Y	0	0	0	0	0
Z	0	0	0	0	0
P_{cat}	0	0	0	0	0
S	0	0	0	0	0
T	0	0	0	0	0

Table 11. Gate count per error correction operation with Bacon-Shor code.

Operation(s)	m=1	m=2	m=3	m=4	m=5
$hCNOT$	29	$3.47e+03$	$5.47e+05$	$1.04e+08$	$2.09e+10$
$vCNOT$	0	0	0	0	0
$hSWAP$	12	$6.85e+03$	$1.66e+06$	$3.51e+08$	$7.24e+10$
$vSWAP$	0	0	0	0	0
H	0	0	0	0	0
$P_{\|+\rangle}$	9	927	$1.52e+05$	$2.94e+07$	$5.94e+09$
$P_{\|0\rangle}$	9	$1.14e+03$	$1.77e+05$	$3.34e+07$	$6.7e+09$
$\mathcal{M}_X$	9	792	$1.34e+05$	$2.67e+07$	$5.43e+09$
$\mathcal{M}_Z$	9	792	$1.34e+05$	$2.67e+07$	$5.43e+09$
X	1	82	$1.44e+04$	$2.87e+06$	$5.85e+08$
Y	0	0	0	0	0
Z	1	82	$1.44e+04$	$2.87e+06$	$5.85e+08$
P_{cat}	0	0	0	0	0
S	0	0	0	0	0
T	0	0	0	0	0

Table 12. Gate count per error correction operation with Knill/Steane code.

Operation(s)	m=1	m=2	m=3	m=4	m=5
$hCNOT$	14	$1.12e+03$	$7.3e+04$	$4.98e+06$	$3.39e+08$
$vCNOT$	28	$1.06e+03$	$7.33e+04$	$4.98e+06$	$3.39e+08$
$hSWAP$	18	$1.85e+03$	$1.38e+05$	$9.51e+06$	$6.5e+08$
$vSWAP$	15	$2.22e+03$	$1.43e+05$	$9.63e+06$	$6.53e+08$
H	7	28	112	448	$1.79e+03$
$P_{\|+\rangle}$	8	696	$4.77e+04$	$3.25e+06$	$2.21e+08$
$P_{\|0\rangle}$	12	696	$4.77e+04$	$3.25e+06$	$2.21e+08$
$\mathcal{M}_X$	20	736	$4.79e+04$	$3.25e+06$	$2.21e+08$
$\mathcal{M}_Z$	0	656	$4.76e+04$	$3.25e+06$	$2.21e+08$
X	0	0	0	0	0
Y	0	0	0	0	0
Z	0	0	0	0	0
P_{cat}	0	0	0	0	0
S	0	0	0	0	0

T	0	0	0	0	0

Table 13. List of abbreviations.

Algorithms	**Technologies**	**Control**
Binary Welded Tree: BWT	Quantum Dots: DOT	Primitive: PRI
Boolean Formula: BFA	Neutral Atoms: NEU	Optimal: OPT
Class Number: CNA	Photonics I: PH1	Solovay Kitaev: SOK
Ground State Est.: GSE	Photonics II: PH2	Trotter: TRO
Quant. Linear Syst.: QLS	Superconductors: SUP	Dynamically Cor. Gates: DCG
Shortest Vector: SVP	Ion Traps: TRA	
Triangle Finding: TFP		
Error Correction	**Syndrome Extraction**	
Steane: STE	Steane: STE	
Bacon-Shor: BSH	Shor: SHO	
Knill's scheme: C46	Knill: KNI	
Surface code: TOP		

Table 14. Final resource estimates.

Algorithm	**Technology**	**Control**	**QECC**	**Syndrome Measurement**	**# Qubits**	**# Gates**	**Running Time**	**Code distance or concatenations**
BWT	NEU	OPT	TOP	KNI	$9.33e+10$	$6.08e+24$	$5.69e+18$	83
BWT	NEU	OPT	TOP	SHO	$1.87e+11$	$4.25e+25$	$6.06e+18$	83
BWT	NEU	OPT	TOP	STE	$6.22e+10$	$1.21e+25$	$5.98e+18$	83
BWT	NEU	PRI	TOP	KNI	$8.88e+10$	$5.65e+24$	$5.55e+18$	81
BWT	NEU	PRI	TOP	SHO	$1.78e+11$	$3.95e+25$	$5.92e+18$	81
BWT	NEU	PRI	TOP	STE	$5.92e+10$	$1.13e+25$	$5.84e+18$	81
BWT	NEU	SOK	TOP	KNI	$8.46e+09$	$1.73e+23$	$2.05e+18$	25
BWT	NEU	SOK	TOP	SHO	$1.69e+10$	$1.2e+24$	$2.39e+18$	25
BWT	NEU	SOK	TOP	STE	$5.64e+09$	$3.43e+23$	$2.31e+18$	25
BWT	NEU	TRO	TOP	KNI	$1.04e+11$	$1.97e+24$	$2.08e+18$	23
BWT	NEU	TRO	TOP	SHO	$2.07e+11$	$1.36e+25$	$2.53e+18$	23
BWT	NEU	TRO	TOP	STE	$6.9e+10$	$3.89e+24$	$2.43e+18$	23
BWT	PH2	PRI	TOP	KNI	$3.25e+10$	$1.31e+24$	$3.46e+15$	49
BWT	PH2	PRI	TOP	SHO	$6.5e+10$	$9.02e+24$	$4.4e+15$	49
BWT	PH2	PRI	TOP	STE	$2.17e+10$	$2.58e+24$	$4.33e+15$	49
BWT	SUP	OPT	C46	STE	$1.54e+13$	$1.06e+45$	$1.12e+19$	6

BWT	SUP	OPT	TOP	KNI	$5.66e+10$	$7.87e+23$	$2.69e+15$	17
BWT	SUP	OPT	TOP	SHO	$1.13e+11$	$5.41e+24$	$3.58e+15$	17
BWT	SUP	OPT	TOP	STE	$3.77e+10$	$1.56e+24$	$3.16e+15$	17
BWT	SUP	PRI	BSH	STE	$1.26e+12$	$6.87e+32$	$4.67e+18$	5
BWT	SUP	PRI	C46	STE	$9.87e+08$	$2.42e+27$	$8.09e+15$	3
BWT	SUP	PRI	STE	STE	$6.46e+09$	$2.01e+30$	$2.35e+18$	4
BWT	SUP	PRI	TOP	KNI	$9.59e+09$	$5.57e+22$	$9.57e+14$	7
BWT	SUP	PRI	TOP	SHO	$1.92e+10$	$3.81e+23$	$1.23e+15$	7
BWT	SUP	PRI	TOP	STE	$6.39e+09$	$1.09e+23$	$1.19e+15$	7
BWT	TRA	DCG	TOP	KNI	$8.45e+10$	$6.11e+24$	$1.98e+20$	79
BWT	TRA	DCG	TOP	SHO	$1.69e+11$	$3.94e+25$	$3.53e+20$	79
BWT	TRA	DCG	TOP	STE	$5.63e+10$	$1.13e+25$	$3.45e+20$	79
BWT	TRA	OPT	BSH	STE	$1.07e+07$	$1.93e+24$	$2.35e+18$	2
BWT	TRA	OPT	C46	STE	$4.1e+07$	$3.32e+21$	$1.4e+18$	2
BWT	TRA	OPT	STE	STE	$1.35e+08$	$7.31e+24$	$1.77e+20$	3
BWT	TRA	OPT	TOP	KNI	$4.89e+09$	$2.26e+22$	$1.66e+18$	5
BWT	TRA	OPT	TOP	SHO	$9.78e+09$	$1.47e+23$	$2.6e+18$	5
BWT	TRA	OPT	TOP	STE	$3.26e+09$	$4.21e+22$	$2.55e+18$	5
BWT	TRA	PRI	BSH	STE	$1.07e+07$	$1.96e+24$	$2.35e+18$	2
BWT	TRA	PRI	C46	STE	$4.1e+07$	$1.54e+22$	$1.4e+18$	2
BWT	TRA	PRI	STE	STE	$2.8e+06$	$1.39e+23$	$6.56e+18$	2
BWT	TRA	PRI	TOP	KNI	$2.63e+10$	$7.55e+22$	$1.02e+18$	3
BWT	TRA	PRI	TOP	SHO	$5.26e+10$	$4.83e+23$	$1.59e+18$	3
BWT	TRA	PRI	TOP	STE	$1.75e+10$	$1.39e+23$	$1.56e+18$	3
BFA	NEU	OPT	TOP	KNI	$1.88e+10$	$4.79e+40$	$2.22e+35$	175
BFA	NEU	OPT	TOP	SHO	$3.76e+10$	$3.35e+41$	$2.37e+35$	175
BFA	NEU	OPT	TOP	STE	$1.25e+10$	$9.57e+40$	$2.34e+35$	175
BFA	NEU	PRI	TOP	KNI	$1.88e+10$	$4.79e+40$	$2.22e+35$	175
BFA	NEU	PRI	TOP	SHO	$3.76e+10$	$3.35e+41$	$2.37e+35$	175
BFA	NEU	PRI	TOP	STE	$1.25e+10$	$9.57e+40$	$2.34e+35$	175
BFA	NEU	SOK	TOP	KNI	$1.86e+09$	$1.5e+39$	$8.13e+34$	55
BFA	NEU	SOK	TOP	SHO	$3.72e+09$	$1.05e+40$	$9.52e+34$	55
BFA	NEU	SOK	TOP	STE	$1.24e+09$	$3e+39$	$9.19e+34$	55
BFA	NEU	TRO	TOP	KNI	$4.66e+09$	$3.49e+39$	$8.23e+34$	51
BFA	NEU	TRO	TOP	SHO	$9.31e+09$	$2.44e+40$	$1.01e+35$	51
BFA	NEU	TRO	TOP	STE	$3.1e+09$	$6.96e+39$	$9.68e+34$	51
BFA	PH2	PRI	TOP	KNI	$7.03e+09$	$1.1e+40$	$1.34e+32$	107
BFA	PH2	PRI	TOP	SHO	$1.41e+10$	$7.66e+40$	$1.73e+32$	107
BFA	PH2	PRI	TOP	STE	$4.69e+09$	$2.19e+40$	$1.7e+32$	107
BFA	SUP	OPT	C46	STE	$4.02e+14$	$4.68e+63$	$1.48e+36$	7
BFA	SUP	OPT	TOP	KNI	$2.45e+09$	$1.32e+39$	$1.04e+32$	37
BFA	SUP	OPT	TOP	SHO	$4.9e+09$	$9.21e+39$	$1.41e+32$	37
BFA	SUP	OPT	TOP	STE	$1.63e+09$	$2.63e+39$	$1.24e+32$	37
BFA	SUP	PRI	BSH	STE	$3.99e+15$	$1.16e+56$	$8.91e+36$	7
BFA	SUP	PRI	C46	STE	$2.57e+10$	$1.08e+46$	$1.07e+33$	4
BFA	SUP	PRI	STE	STE	$3.22e+13$	$1.13e+54$	$1.92e+37$	6
BFA	SUP	PRI	TOP	KNI	$5.17e+08$	$1.28e+38$	$4.11e+31$	17
BFA	SUP	PRI	TOP	SHO	$1.03e+09$	$8.96e+38$	$5.34e+31$	17

BFA	SUP	PRI	TOP	STE	$3.45e+08$	$2.56e+38$	$5.2e+31$	17
BFA	TRA	DCG	TOP	KNI	$1.8e+10$	$4.47e+40$	$6.81e+36$	171
BFA	TRA	DCG	TOP	SHO	$3.59e+10$	$3.12e+41$	$1.31e+37$	171
BFA	TRA	DCG	TOP	STE	$1.2e+10$	$8.91e+40$	$1.28e+37$	171
BFA	TRA	OPT	BSH	STE	$3.39e+10$	$5.94e+43$	$4.52e+36$	4
BFA	TRA	OPT	C46	STE	$1.03e+09$	$1.66e+40$	$1.84e+35$	3
BFA	TRA	OPT	STE	STE	$1.4e+10$	$8.43e+44$	$5.31e+37$	4
BFA	TRA	OPT	TOP	KNI	$2.17e+08$	$3.55e+37$	$5.94e+34$	11
BFA	TRA	OPT	TOP	SHO	$4.33e+08$	$2.46e+38$	$9.85e+34$	11
BFA	TRA	OPT	TOP	STE	$1.44e+08$	$7.02e+37$	$9.66e+34$	11
BFA	TRA	PRI	BSH	STE	$6.92e+08$	$1.71e+42$	$2.91e+35$	3
BFA	TRA	PRI	C46	STE	$1.03e+09$	$2.5e+41$	$1.84e+35$	3
BFA	TRA	PRI	STE	STE	$2.91e+08$	$1.59e+43$	$1.94e+36$	3
BFA	TRA	PRI	TOP	KNI	$9.43e+08$	$9.99e+37$	$3.84e+34$	7
BFA	TRA	PRI	TOP	SHO	$1.89e+09$	$6.87e+38$	$6.33e+34$	7
BFA	TRA	PRI	TOP	STE	$6.29e+08$	$1.96e+38$	$6.2e+34$	7
CNA	NEU	OPT	TOP	KNI	$5.1e+23$	$4.01e+45$	$6.87e+26$	119
CNA	NEU	OPT	TOP	SHO	$1.02e+24$	$2.81e+46$	$7.32e+26$	119
CNA	NEU	OPT	TOP	STE	$3.4e+23$	$8.02e+45$	$7.23e+26$	119
CNA	NEU	PRI	TOP	KNI	$4.93e+23$	$3.81e+45$	$6.76e+26$	117
CNA	NEU	PRI	TOP	SHO	$9.86e+23$	$2.67e+46$	$7.2e+26$	117
CNA	NEU	PRI	TOP	STE	$3.29e+23$	$7.62e+45$	$7.11e+26$	117
CNA	NEU	SOK	TOP	KNI	$4.93e+22$	$1.24e+44$	$2.52e+26$	37
CNA	NEU	SOK	TOP	SHO	$9.86e+22$	$8.6e+44$	$2.94e+26$	37
CNA	NEU	SOK	TOP	STE	$3.29e+22$	$2.46e+44$	$2.84e+26$	37
CNA	NEU	TRO	TOP	KNI	$4.41e+22$	$1.05e+44$	$2.62e+26$	35
CNA	NEU	TRO	TOP	SHO	$8.82e+22$	$7.31e+44$	$3.21e+26$	35
CNA	NEU	TRO	TOP	STE	$2.94e+22$	$2.09e+44$	$3.07e+26$	35
CNA	PH2	PRI	TOP	KNI	$1.81e+23$	$8.75e+44$	$4.15e+23$	71
CNA	PH2	PRI	TOP	SHO	$3.63e+23$	$6.07e+45$	$5.3e+23$	71
CNA	PH2	PRI	TOP	STE	$1.21e+23$	$1.73e+45$	$5.21e+23$	71
CNA	SUP	OPT	C46	STE	$2.2e+27$	$5.87e+56$	$1.02e+28$	7
CNA	SUP	OPT	TOP	KNI	$2.25e+22$	$3.8e+43$	$3.28e+23$	25
CNA	SUP	OPT	TOP	SHO	$4.5e+22$	$2.64e+44$	$4.39e+23$	25
CNA	SUP	OPT	TOP	STE	$1.5e+22$	$7.56e+43$	$3.87e+23$	25
CNA	SUP	PRI	BSH	STE	$2.6e+27$	$2.4e+58$	$5.29e+27$	6
CNA	SUP	PRI	C46	STE	$5.63e+21$	$7.68e+46$	$6.62e+23$	3
CNA	SUP	PRI	STE	STE	$4.78e+25$	$1.22e+56$	$5.19e+27$	5
CNA	SUP	PRI	TOP	KNI	$4.36e+21$	$3.26e+42$	$1.24e+23$	11
CNA	SUP	PRI	TOP	SHO	$8.71e+21$	$2.26e+43$	$1.6e+23$	11
CNA	SUP	PRI	TOP	STE	$2.9e+21$	$6.46e+42$	$1.56e+23$	11
CNA	TRA	DCG	TOP	KNI	$4.76e+23$	$3.96e+45$	$2.27e+28$	115
CNA	TRA	DCG	TOP	SHO	$9.52e+23$	$2.64e+46$	$4.2e+28$	115
CNA	TRA	DCG	TOP	STE	$3.17e+23$	$7.55e+45$	$4.1e+28$	115
CNA	TRA	OPT	BSH	STE	$2.21e+22$	$1.22e+46$	$2.68e+27$	3
CNA	TRA	OPT	C46	STE	$5.63e+21$	$1.26e+45$	$1.26e+27$	3
CNA	TRA	OPT	STE	STE	$2.07e+22$	$9.11e+46$	$1.44e+28$	3
CNA	TRA	OPT	TOP	KNI	$1.76e+21$	$9.01e+41$	$1.85e+26$	7

CNA	TRA	OPT	TOP	SHO	$3.53e+21$	$6.04e+42$	$2.97e+26$	7
CNA	TRA	OPT	TOP	STE	$1.18e+21$	$1.73e+42$	$2.92e+26$	7
CNA	TRA	PRI	BSH	STE	$4.5e+20$	$3.61e+44$	$1.72e+26$	2
CNA	TRA	PRI	C46	STE	$2.25e+20$	$1.53e+44$	$1.14e+26$	2
CNA	TRA	PRI	STE	STE	$4.32e+20$	$1.73e+45$	$5.29e+26$	2
CNA	TRA	PRI	TOP	KNI	$9e+20$	$3.33e+41$	$1.34e+26$	5
CNA	TRA	PRI	TOP	SHO	$1.8e+21$	$2.22e+42$	$2.14e+26$	5
CNA	TRA	PRI	TOP	STE	$6e+20$	$6.35e+41$	$2.1e+26$	5
GSE	NEU	OPT	TOP	KNI	$1.93e+09$	$6.18e+32$	$2.8e+28$	129
GSE	NEU	OPT	TOP	SHO	$3.86e+09$	$4.32e+33$	$2.98e+28$	129
GSE	NEU	OPT	TOP	STE	$1.29e+09$	$1.23e+33$	$2.94e+28$	129
GSE	NEU	PRI	TOP	KNI	$1.93e+09$	$6.18e+32$	$2.8e+28$	129
GSE	NEU	PRI	TOP	SHO	$3.86e+09$	$4.32e+33$	$2.98e+28$	129
GSE	NEU	PRI	TOP	STE	$1.29e+09$	$1.23e+33$	$2.94e+28$	129
GSE	NEU	SOK	TOP	KNI	$1.95e+08$	$2.05e+31$	$1.06e+28$	41
GSE	NEU	SOK	TOP	SHO	$3.89e+08$	$1.42e+32$	$1.23e+28$	41
GSE	NEU	SOK	TOP	STE	$1.3e+08$	$4.07e+31$	$1.19e+28$	41
GSE	NEU	TRO	TOP	KNI	$1.42e+09$	$1.36e+32$	$1.05e+28$	37
GSE	NEU	TRO	TOP	SHO	$2.84e+09$	$9.44e+32$	$1.28e+28$	37
GSE	NEU	TRO	TOP	STE	$9.48e+08$	$2.7e+32$	$1.23e+28$	37
GSE	PH2	PRI	TOP	KNI	$7.23e+08$	$1.48e+32$	$1.76e+25$	79
GSE	PH2	PRI	TOP	SHO	$1.45e+09$	$1.02e+33$	$2.25e+25$	79
GSE	PH2	PRI	TOP	STE	$4.82e+08$	$2.93e+32$	$2.21e+25$	79
GSE	SUP	OPT	C46	STE	$3.73e+14$	$1.27e+58$	$3.95e+29$	7
GSE	SUP	OPT	TOP	KNI	$8.74e+08$	$6.56e+31$	$1.45e+25$	29
GSE	SUP	OPT	TOP	SHO	$1.75e+09$	$4.52e+32$	$1.94e+25$	29
GSE	SUP	OPT	TOP	STE	$5.83e+08$	$1.3e+32$	$1.71e+25$	29
GSE	SUP	PRI	BSH	STE	$4.8e+13$	$1.22e+45$	$2.29e+29$	6
GSE	SUP	PRI	C46	STE	$2.39e+10$	$2.91e+40$	$2.86e+26$	4
GSE	SUP	PRI	STE	STE	$5.61e+10$	$5.61e+42$	$2.04e+29$	5
GSE	SUP	PRI	TOP	KNI	$1.76e+08$	$5.95e+30$	$5.6e+24$	13
GSE	SUP	PRI	TOP	SHO	$3.51e+08$	$4.09e+31$	$7.19e+24$	13
GSE	SUP	PRI	TOP	STE	$1.17e+08$	$1.17e+31$	$7e+24$	13
GSE	TRA	DCG	TOP	KNI	$1.81e+09$	$6.57e+32$	$9.93e+29$	125
GSE	TRA	DCG	TOP	SHO	$3.62e+09$	$4.24e+33$	$1.77e+30$	125
GSE	TRA	DCG	TOP	STE	$1.21e+09$	$1.21e+33$	$1.74e+30$	125
GSE	TRA	OPT	BSH	STE	$4.08e+08$	$6.82e+33$	$1.16e+29$	3
GSE	TRA	OPT	C46	STE	$9.57e+08$	$3.84e+34$	$4.89e+28$	3
GSE	TRA	OPT	STE	STE	$1.17e+09$	$2.04e+37$	$1.53e+31$	4
GSE	TRA	OPT	TOP	KNI	$8.42e+07$	$2.17e+30$	$9.28e+27$	9
GSE	TRA	OPT	TOP	SHO	$1.68e+08$	$1.42e+31$	$1.47e+28$	9
GSE	TRA	OPT	TOP	STE	$5.61e+07$	$4.08e+30$	$1.44e+28$	9
GSE	TRA	PRI	BSH	STE	$8.33e+06$	$4.89e+31$	$7.47e+27$	2
GSE	TRA	PRI	C46	STE	$3.98e+07$	$7.04e+30$	$4.44e+27$	2
GSE	TRA	PRI	STE	STE	$2.43e+07$	$3.85e+35$	$5.63e+29$	3
GSE	TRA	PRI	TOP	KNI	$3.67e+08$	$5.38e+30$	$5.26e+27$	5
GSE	TRA	PRI	TOP	SHO	$7.34e+08$	$3.5e+31$	$8.26e+27$	5
GSE	TRA	PRI	TOP	STE	$2.45e+08$	$1e+31$	$8.11e+27$	5

QLS	NEU	OPT	TOP	KNI	$1.51e+11$	$1.03e+48$	$5.94e+41$	227
QLS	NEU	OPT	TOP	SHO	$3.02e+11$	$7.19e+48$	$6.33e+41$	227
QLS	NEU	OPT	TOP	STE	$1.01e+11$	$2.05e+48$	$6.25e+41$	227
QLS	NEU	PRI	TOP	KNI	$1.48e+11$	$1e+48$	$5.89e+41$	225
QLS	NEU	PRI	TOP	SHO	$2.97e+11$	$7e+48$	$6.28e+41$	225
QLS	NEU	PRI	TOP	STE	$9.89e+10$	$2e+48$	$6.19e+41$	225
QLS	NEU	SOK	TOP	KNI	$1.48e+10$	$3.23e+46$	$2.2e+41$	71
QLS	NEU	SOK	TOP	SHO	$2.96e+10$	$2.25e+47$	$2.56e+41$	71
QLS	NEU	SOK	TOP	STE	$9.85e+09$	$6.43e+46$	$2.48e+41$	71
QLS	NEU	TRO	TOP	KNI	$1.79e+11$	$3.62e+47$	$2.22e+41$	65
QLS	NEU	TRO	TOP	SHO	$3.58e+11$	$2.51e+48$	$2.71e+41$	65
QLS	NEU	TRO	TOP	STE	$1.19e+11$	$7.18e+47$	$2.6e+41$	65
QLS	PH2	PRI	TOP	KNI	$5.5e+10$	$2.36e+47$	$3.69e+38$	137
QLS	PH2	PRI	TOP	SHO	$1.1e+11$	$1.63e+48$	$4.7e+38$	137
QLS	PH2	PRI	TOP	STE	$3.67e+10$	$4.67e+47$	$4.62e+38$	137
QLS	SUP	OPT	C46	STE	$9.34e+15$	$5.07e+78$	$5.31e+43$	8
QLS	SUP	OPT	TOP	KNI	$1.02e+11$	$1.56e+47$	$2.97e+38$	49
QLS	SUP	OPT	TOP	SHO	$2.03e+11$	$1.07e+48$	$3.95e+38$	49
QLS	SUP	OPT	TOP	STE	$6.78e+10$	$3.09e+47$	$3.49e+38$	49
QLS	SUP	PRI	BSH	STE	$2.38e+15$	$5.39e+61$	$4.28e+43$	7
QLS	SUP	PRI	C46	STE	$2.39e+10$	$1.53e+55$	$3.46e+39$	4
QLS	SUP	PRI	STE	STE	$3.12e+12$	$3.83e+59$	$6.72e+43$	6
QLS	SUP	PRI	TOP	KNI	$1.87e+10$	$1.23e+46$	$1.09e+38$	21
QLS	SUP	PRI	TOP	SHO	$3.73e+10$	$8.48e+46$	$1.4e+38$	21
QLS	SUP	PRI	TOP	STE	$1.24e+10$	$2.43e+46$	$1.36e+38$	21
QLS	TRA	DCG	TOP	KNI	$1.43e+11$	$1.11e+48$	$2.12e+43$	221
QLS	TRA	DCG	TOP	SHO	$2.86e+11$	$7.18e+48$	$3.79e+43$	221
QLS	TRA	DCG	TOP	STE	$9.54e+10$	$2.06e+48$	$3.71e+43$	221
QLS	TRA	OPT	BSH	STE	$2.02e+10$	$6.92e+50$	$2.17e+43$	4
QLS	TRA	OPT	C46	STE	$2.39e+10$	$1.53e+55$	$6.58e+42$	4
QLS	TRA	OPT	STE	STE	$6.5e+10$	$1.39e+54$	$5.07e+45$	5
QLS	TRA	OPT	TOP	KNI	$9.53e+09$	$4.89e+45$	$1.85e+41$	15
QLS	TRA	OPT	TOP	SHO	$1.91e+10$	$3.23e+46$	$2.94e+41$	15
QLS	TRA	OPT	TOP	STE	$6.35e+09$	$9.25e+45$	$2.89e+41$	15
QLS	TRA	PRI	BSH	STE	$4.12e+08$	$4.05e+48$	$1.41e+42$	3
QLS	TRA	PRI	C46	STE	$9.58e+08$	$2.02e+49$	$5.91e+41$	3
QLS	TRA	PRI	STE	STE	$1.35e+09$	$2.63e+52$	$1.86e+44$	4
QLS	TRA	PRI	TOP	KNI	$7.66e+10$	$2.9e+46$	$1.37e+41$	11
QLS	TRA	PRI	TOP	SHO	$1.53e+11$	$1.91e+47$	$2.16e+41$	11
QLS	TRA	PRI	TOP	STE	$5.11e+10$	$5.47e+46$	$2.13e+41$	11
SVP	NEU	OPT	TOP	KNI	$1.8e+25$	$4.63e+50$	$2.25e+30$	153
SVP	NEU	OPT	TOP	SHO	$3.6e+25$	$3.24e+51$	$2.4e+30$	153
SVP	NEU	OPT	TOP	STE	$1.2e+25$	$9.25e+50$	$2.36e+30$	153
SVP	NEU	PRI	TOP	KNI	$1.75e+25$	$4.45e+50$	$2.22e+30$	151
SVP	NEU	PRI	TOP	SHO	$3.5e+25$	$3.11e+51$	$2.37e+30$	151
SVP	NEU	PRI	TOP	STE	$1.17e+25$	$8.89e+50$	$2.33e+30$	151
SVP	NEU	SOK	TOP	KNI	$1.7e+24$	$1.36e+49$	$8.06e+29$	47
SVP	NEU	SOK	TOP	SHO	$3.39e+24$	$9.49e+49$	$9.44e+29$	47

SVP	NEU	SOK	TOP	STE	$1.13e+24$	$2.71e+49$	$9.11e+29$	47
SVP	NEU	TRO	TOP	KNI	$1.56e+24$	$1.2e+49$	$8.43e+29$	45
SVP	NEU	TRO	TOP	SHO	$3.11e+24$	$8.33e+49$	$1.04e+30$	45
SVP	NEU	TRO	TOP	STE	$1.04e+24$	$2.38e+49$	$9.92e+29$	45
SVP	PH2	PRI	TOP	KNI	$6.64e+24$	$1.05e+50$	$1.35e+27$	93
SVP	PH2	PRI	TOP	SHO	$1.33e+25$	$7.3e+50$	$1.74e+27$	93
SVP	PH2	PRI	TOP	STE	$4.43e+24$	$2.09e+50$	$1.71e+27$	93
SVP	SUP	OPT	C46	STE	$4.69e+28$	$5.36e+61$	$1.93e+31$	7
SVP	SUP	OPT	TOP	KNI	$8.36e+23$	$4.66e+48$	$1.08e+27$	33
SVP	SUP	OPT	TOP	SHO	$1.67e+24$	$3.25e+49$	$1.46e+27$	33
SVP	SUP	OPT	TOP	STE	$5.58e+23$	$9.3e+48$	$1.28e+27$	33
SVP	SUP	PRI	BSH	STE	$5.54e+28$	$8.01e+62$	$8.3e+30$	6
SVP	SUP	PRI	C46	STE	$3e+24$	$2.35e+54$	$1.4e+28$	4
SVP	SUP	PRI	STE	STE	$4.89e+28$	$2.29e+64$	$2.57e+32$	6
SVP	SUP	PRI	TOP	KNI	$1.73e+23$	$4.39e+47$	$4.22e+26$	15
SVP	SUP	PRI	TOP	SHO	$3.46e+23$	$3.06e+48$	$5.47e+26$	15
SVP	SUP	PRI	TOP	STE	$1.15e+23$	$8.76e+47$	$5.32e+26$	15
SVP	TRA	DCG	TOP	KNI	$1.71e+25$	$4.35e+50$	$6.98e+31$	149
SVP	TRA	DCG	TOP	SHO	$3.41e+25$	$3.01e+51$	$1.34e+32$	149
SVP	TRA	DCG	TOP	STE	$1.14e+25$	$8.6e+50$	$1.3e+32$	149
SVP	TRA	OPT	BSH	STE	$4.71e+23$	$4.08e+50$	$4.19e+30$	3
SVP	TRA	OPT	C46	STE	$1.2e+23$	$5.09e+49$	$2.39e+30$	3
SVP	TRA	OPT	STE	STE	$2.12e+25$	$1.71e+55$	$7.09e+32$	4
SVP	TRA	OPT	TOP	KNI	$6.22e+22$	$9.83e+46$	$5.72e+29$	9
SVP	TRA	OPT	TOP	SHO	$1.24e+23$	$6.74e+47$	$9.41e+29$	9
SVP	TRA	OPT	TOP	STE	$4.15e+22$	$1.93e+47$	$9.23e+29$	9
SVP	TRA	PRI	BSH	STE	$4.71e+23$	$3.75e+52$	$4.19e+30$	3
SVP	TRA	PRI	C46	STE	$4.8e+21$	$6.21e+48$	$2.17e+29$	2
SVP	TRA	PRI	STE	STE	$4.42e+23$	$3.23e+53$	$2.6e+31$	3
SVP	TRA	PRI	TOP	KNI	$3.76e+22$	$4.66e+46$	$4.49e+29$	7
SVP	TRA	PRI	TOP	SHO	$7.53e+22$	$3.19e+47$	$7.36e+29$	7
SVP	TRA	PRI	TOP	STE	$2.51e+22$	$9.12e+46$	$7.22e+29$	7
TFP	NEU	OPT	TOP	KNI	$1.44e+14$	$5.82e+31$	$3.53e+22$	91
TFP	NEU	OPT	TOP	SHO	$2.87e+14$	$4.07e+32$	$3.77e+22$	91
TFP	NEU	OPT	TOP	STE	$9.58e+13$	$1.16e+32$	$3.72e+22$	91
TFP	NEU	PRI	TOP	KNI	$1.37e+14$	$5.44e+31$	$3.46e+22$	89
TFP	NEU	PRI	TOP	SHO	$2.75e+14$	$3.81e+32$	$3.68e+22$	89
TFP	NEU	PRI	TOP	STE	$9.16e+13$	$1.09e+32$	$3.64e+22$	89
TFP	NEU	SOK	TOP	KNI	$1.46e+13$	$1.94e+30$	$1.33e+22$	29
TFP	NEU	SOK	TOP	SHO	$2.92e+13$	$1.35e+31$	$1.56e+22$	29
TFP	NEU	SOK	TOP	STE	$9.73e+12$	$3.85e+30$	$1.5e+22$	29
TFP	NEU	TRO	TOP	KNI	$1.27e+13$	$1.57e+30$	$1.36e+22$	27
TFP	NEU	TRO	TOP	SHO	$2.53e+13$	$1.09e+31$	$1.67e+22$	27
TFP	NEU	TRO	TOP	STE	$8.43e+12$	$3.12e+30$	$1.6e+22$	27
TFP	PH2	PRI	TOP	KNI	$5.25e+13$	$1.32e+31$	$2.16e+19$	55
TFP	PH2	PRI	TOP	SHO	$1.05e+14$	$9.15e+31$	$2.76e+19$	55
TFP	PH2	PRI	TOP	STE	$3.5e+13$	$2.62e+31$	$2.71e+19$	55
TFP	SUP	OPT	C46	STE	$4.24e+16$	$1.52e+46$	$5.97e+22$	6

TFP	SUP	OPT	TOP	KNI	$6.27e+12$	$5.41e+29$	$1.68e+19$	19
TFP	SUP	OPT	TOP	SHO	$1.25e+13$	$3.75e+30$	$2.24e+19$	19
TFP	SUP	OPT	TOP	STE	$4.18e+12$	$1.08e+30$	$1.98e+19$	19
TFP	SUP	PRI	BSH	STE	$1.25e+18$	$7.34e+44$	$3.37e+23$	6
TFP	SUP	PRI	C46	STE	$2.71e+12$	$2.41e+33$	$4.32e+19$	3
TFP	SUP	PRI	STE	STE	$2.3e+16$	$3.83e+42$	$3.38e+23$	5
TFP	SUP	PRI	TOP	KNI	$1.41e+12$	$5.79e+28$	$6.83e+18$	9
TFP	SUP	PRI	TOP	SHO	$2.81e+12$	$4.01e+29$	$8.8e+18$	9
TFP	SUP	PRI	TOP	STE	$9.37e+11$	$1.15e+29$	$8.57e+18$	9
TFP	TRA	DCG	TOP	KNI	$1.31e+14$	$5.57e+31$	$1.16e+24$	87
TFP	TRA	DCG	TOP	SHO	$2.63e+14$	$3.71e+32$	$2.13e+24$	87
TFP	TRA	DCG	TOP	STE	$8.76e+13$	$1.06e+32$	$2.09e+24$	87
TFP	TRA	OPT	BSH	STE	$2.17e+11$	$1.2e+29$	$1.09e+22$	2
TFP	TRA	OPT	C46	STE	$1.08e+11$	$5.24e+28$	$7.44e+21$	2
TFP	TRA	OPT	STE	STE	$9.99e+12$	$2.86e+33$	$9.36e+23$	3
TFP	TRA	OPT	TOP	KNI	$4.34e+11$	$1.08e+28$	$9.01e+21$	5
TFP	TRA	OPT	TOP	SHO	$8.68e+11$	$7.19e+28$	$1.44e+22$	5
TFP	TRA	OPT	TOP	STE	$2.89e+11$	$2.06e+28$	$1.41e+22$	5
TFP	TRA	PRI	BSH	STE	$2.17e+11$	$1.1e+31$	$1.09e+22$	2
TFP	TRA	PRI	C46	STE	$1.08e+11$	$4.81e+30$	$7.44e+21$	2
TFP	TRA	PRI	STE	STE	$2.08e+11$	$5.43e+31$	$3.45e+22$	2
TFP	TRA	PRI	TOP	KNI	$4.34e+11$	$1.08e+28$	$9.01e+21$	5
TFP	TRA	PRI	TOP	SHO	$8.68e+11$	$7.19e+28$	$1.44e+22$	5
TFP	TRA	PRI	TOP	STE	$2.89e+11$	$2.06e+28$	$1.41e+22$	5

16 Conclusion

This report quantifies the resources needed to run a variety of quantum programs on quantum computers with realistic properties. We quantify the number of physical qubits required to run each program, the execution time on each of the physical technologies of choice, the probability of success of the computation, as well as gate count for each quantum gate type. In the course of performing this resource estimation, we made a number of interesting observations. One of our most interesting observations is the fact that the amount of resources needed to perform topological versus concatenated quantum error correction is comparable. This is surprising because the nature of the codes and the model of quantum computation is very different. However, we still believe that topological error correction is superior in systems with a large number of qubits. Due to its much higher error-correction threshold, topological error correction is able to protect quantum information in systems that cannot be protected by concatenated codes.

17 Acknowledgements

This work was supported by the IARPA QCS program (document numbers D11PC20165 and D11PC20167). The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DoI/NBC, or the U.S. Government.

This work wouldn't be possible without the contributions of the many students and postdocs in the IARPA QCS project who analyzed the properties of the algorithms and quantum technologies that are used by the QuRE toolbox. The description of concatenated codes and some of the figures in this report are based on unpublished work of Tzvetan Metodi, Mark Rubin, Venkat Chandar, John Cortese, Anand Ganti, Andrew Landahl, and Uzoma Onunkwo. We also appreciate the many fruitful discussions with Kenneth Brown, Todd Brun, Austin Fowler, David Hocker, Daniel Lidar, Massoud Pedram, and Robert Rausendorf that helped us to improve our resource estimation methodology.

References

[1] A. M. Steane. Error correcting codes in quantum theory. *Phys. Rev. Lett.*, 77:793–797, Jul 1996.

[2] A. M. Steane. Efficient fault-tolerant quantum computing. *Nature*, 399:124–126, May 1999.

[3] David Hocker, Herschel Rabitz, Yi-Cong Zheng, Todd Brun, Alireza Shafaei, and Massoud Pedram. Evaluation of Quantum Control Protocols for Physical Machine Descriptions. *IARPA QCS Project report*, 2012.

[4] Amlan Chakrabarti, ChiaChun Lin, and Niraj K. Jha. Analysis of the Binary Welded Tree Algorithm. *IARPA QCS Project report*, 2012.

[5] Mohammad Javad Dousti and Massoud Pedram. Implementation of Boolean Formula Algorithm. *IARPA QCS Project report*, 2012.

[6] Oana Theogarajan, Arvin Faruque, Guoping Long, and Fred Chong. Analysis of the Ground State Estimation Algorithm. *IARPA QCS Project report*, 2012.

[7] Lukas Svec and Seth Vanderwilt. GFI- QLSA Progress Overview. *IARPA QCS Project report*, 2012.

[8] Oana Theogarajan, Arvin Faruque, Guoping Long, and Fred Chong. Preliminary Implementation of the Shortest Vector Problem Algorithm. *IARPA QCS Project report*, 2012.

[9] John Black. Preliminary Resource Estimates for Hallgren's Quantum Class Number Algorithm. *IARPA QCS Project report*, 2012.

[10] Chen-Fu Chiang. Analysis of an Implementation of a Quantum Algorithm for the Triangle Finding Problem. *IARPA QCS Project report*, 2012.

[11] Krysta Svore, David DiVincenzo, and Barbara Terhal. Noise Threshold for a Fault-Tolerant Two-Dimensional Lattice Architecture. *arXiv:quant-ph/0604090v3*, 2006.

[12] Federico M. Spedalieri and Vwani P. Roychowdhury. Latency in local, two-dimensional, fault-tolerant quantum computing. *Quantum Info. Comput.*, 9(7):666–682, July 2009.

[13] M. Saffman, T. G. Walker, and K. Mølmer. Quantum information with rydberg atoms. *Rev. Mod. Phys.*, 82:2313–2363, Aug 2010.

[14] Alexandre Blais, Jay Gambetta, A. Wallraff, D. I. Schuster, S. M. Girvin, M. H. Devoret, and R. J. Schoelkopf. Quantum-information processing with circuit quantum electrodynamics. *Phys. Rev. A*, 75:032329, Mar 2007.

[15] John Martinis. Quantum-information processing with circuit quantum electrodynamics. *Quant. Inf. Processing*, 8:81 – 103, Jun 2009.

[16] M. D. Barrett, T. Schaetz, J. Chiaverini, D. Leibfried, J. Britton, W. M. Itano, J. D. Jost, E. Knill, C. Langer, R. Ozeri, and D.J. Wineland. Quantum information processing with trapped ions. In *Proceedings of ICAP*, pages 350 – 358, 2004.

[17] E. Knill, R. Laflamme, and G. J. Milburn. A scheme for efficient quantum computation with linear optics. *Nature*, 409:46–52, January 2001.

[18] Xubo Zou, ShengLi Zhang, Ke Li, and Guangcan Guo. Linear optical implementation of the two-qubit controlled phase gate with conventional photon detectors. *Phys. Rev. A*, 75:034302, Mar 2007.

[19] W J Munro, Kae Nemoto, T P Spiller, S D Barrett, Pieter Kok, and R G Beausoleil. Efficient optical quantum information processing. *Journal of Optics B: Quantum and Semiclassical Optics*, 7(7):S135, 2005.

[20] J. M. Taylor, J. R. Petta, A. C. Johnson, A. Yacoby, C. M. Marcus, and MD Lukin. Relaxation, dephasing, and quantum control of electron spins in double quantum dots. 76(3), 2007. Phys. Rev. B.

[21] Andrew Childs, Richard Cleve, Enrico Deotto, Edward Farhi, Sam Gutmann, and Daniel Spielman. Exponential algorithmic speedup by quantum walk. *arXiv: quant-ph/0209131*, 2002.

[22] Andris Ambainis, Andrew M. Childs, Ben W. Reichardt, Robert Spalek, and Shengyu Zhang. Any and-or formula of size n can be evaluated in time n^1/2 + o(1) on a quantum computer. In *Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science*, pages 363–372, 2007.

[23] James Whitfield, Jacob Biamonte, and Alan Aspuru-Guzik. Simulation of electronic structure Hamiltonians using quantum computers. *Molecular Physics: An International Journal at the Interface Between Chemistry and Physics*, 109(5):735–750, 2011.

[24] Aram Harrow, Avinatan Hassidim, and Seth Lloyd. Quantum algorithm for solving linear systems of equations. *Phys. Rev. Lett.*, 103:150502, Jul 1999.

[25] Oded Regev. Quantum computation and lattice problems. *SIAM J. Comput.*, 33(3):738–760, March 2004.

[26] Sean Hallgren. Polynomial-time quantum algorithms for Pell's equation and the principal ideal problem. *Journal of ACM*, 54(1):4:1–4:19, March 2007.

[27] Frederic Magniez, Miklos Santha, and Mario Szegedy. Quantum algorithms for the triangle problem. *arXiv: quant-ph/0310134*, 2003.

[28] Alex Bocharov and Krysta Svore. A Depth-Optimal Canonical Form for Single-qubit Quantum Circuits. *arXiv: quant-ph/1206.3223*, 2012.

[29] A. Yu. Kitaev, A. H. Shen, and M. N. Vyalyi. *Classical and Quantum Computation.* American Mathematical Society, Boston, MA, USA, 2002.

[30] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information.* Cambridge University Press, 2000.

[31] Xinlan Zhou, Debbie W. Leung, and Isaac L. Chuang. Methodology for quantum logic gate construction. *Phys. Rev. A*, 62:052316, Oct 2000.

[32] Panos Aliferis, Daniel Gottesman, and John Preskill. Quantum accuracy threshold for concatenated distance-3 codes. *ArXiv quant-ph/0504218v3*, 2005.

[33] E. Knill. Quantum computing with realistically noisy devices. *Nature*, 434:39–44, 2005.

[34] Ching-Yi Lai, Gerardo Paz, Martin Suchara, and Todd A. Brun. Analysis of the knill c4/c6 code in atwo-dimensional architecture. 2013.

[35] Krysta M. Svore, Barbara M. Terhal, and David P. DiVincenzo. Local fault-tolerant quantum computation. *arXiv:quant-ph/0410047v2*, 2004.

[36] Eric Dennis, Alexei Kitaev, Andrew Landahl, and John Preskill. Topological quantum memory. *J. Math. Phys.*, 43(9):4452, Sep 2002.

[37] A. Fowler, A. Stephens, and P. Groszkowski. High threshold universal quantum computation on the surface code. *Phys. Rev. A*, 80:052312, 2009.

[38] A. Steane. Overhead and noise threshold of fault-tolerant quantum error correction. *Phys. Rev. A*, 68:042322, 2003.

[39] P. Aliferis. Level Reduction and the Quantum Threshold Theorem. *arXiv:quant-ph/0703230v2*, 2007.

[40] N. Jones, R. Van Meter, A. Fowler, P. McMahon, J. Kim, T. Ladd, and Y. Yamamoto. Layered architecture for quantum computing. *arXiv:quant-ph/1010.5022v3*, 2012.

[41] S. Bravyi and A. Kitaev. Universal quantum computation with ideal clifford gates and noisy ancillas. *Phys. Rev. A*, 71:022316, 2005.

Appendix A

17.1 Implementation of the tile operations of the C_4 code

In the following, "$a \rightarrow b$" means applying a CNOT gate with a being the control qubit and b being the target qubit. "$a \Leftrightarrow b$" means applying a SWAP gate on qubits a and b.
zED

Time step 1:

$$
\begin{array}{ccccc}
O & O & O & O & O \\
O & d_1 & O & O & d_3 \\
O & P_{|+\rangle}(a_1) & P_{|+\rangle}(a_5) & P_{|0\rangle}(a_7) & P_{|+\rangle}(a_3) \\
O & P_{|0\rangle}(a_2) & P_{|+\rangle}(a_6) & P_{|0\rangle}(a_8) & P_{|0\rangle}(a_4) \\
O & d_2 & O & O & d_4
\end{array}
$$

Time step 2:

$$
\begin{tikzcd}
O & O & O & O & O \\
O & d_1 & O & O & d_3 \\
O & a_1 \arrow[d] & a_5 \arrow[r] & a_7 & a_3 \arrow[d] \\
O & a_2 & a_6 \arrow[r] & a_8 & a_4 \\
O & d_2 & O & O & d_4.
\end{tikzcd}
$$

Time step 3:

$$
\begin{tikzcd}
O & O & O & O & O \\
O & d_1 & O & O & d_3 \\
O & a_1 \arrow[r] & a_5 & a_7 & a_3 \arrow[l] \\
O & a_2 \arrow[r] & a_6 & a_8 & a_4 \arrow[l] \\
O & d_2 & O & O & d_4
\end{tikzcd}
$$

Time step 4:

$$
\begin{tikzcd}
O & O & O & O & O \\
O & d_1 \arrow[d] & O & O & d_3 \arrow[d] \\
O & a_1 & a_5 & a_7 & a_3 \\
O & a_2 & a_6 & a_8 & a_4 \\
O & d_2 \arrow[u] & O & O & d_4 \arrow[u]
\end{tikzcd}
$$

Time step 5:

$$\begin{array}{ccccc}
O & O & O & O & O \\
O & M_X(d_1) & O & O & M_X(d_3) \\
O & M_Z(a_1) & a_5 & a_7 & M_Z(a_3) \\
O & M_Z(a_2) & a_6 & a_8 & M_Z(a_4) \\
O & M_X(d_2) & O & O & M_X(d_4)
\end{array}$$

We choose the index such that a quantum teleportation occurs on the qubits d_i, a_i, a_{i+4} for $i = 1, 2, 3, 4$. Observe that the data qubits d_1, d_2, d_3, d_4 are transferred to the center after teleportation and no SWAPs are needed here. However, the error detection ED_+ for structure II needs two SWAPs and it takes one more step. Its first time step is initialized as follows:

$$\begin{array}{ccccc}
O & O & O & O & O \\
O & O & P_{|+\rangle}(a_1) & P_{|0\rangle}(a_3) & O \\
O & P_{|+\rangle}(a_5) & d_1 & d_3 & P_{|+\rangle}(a_7) \\
O & P_{|0\rangle}(a_6) & d_2 & d_4 & P_{|0\rangle}(a_8) \\
O & O & P_{|+\rangle}(a_2) & P_{|0\rangle}(a_4) & O
\end{array}.$$

In addition, applying the Pauli operators X or Z to complete the teleportation may takes one or two more steps, but this is not shown. The ED_0 for structure 1 at time step 1 is as follows and the rest of the steps are similar to those of the ED_+:

$$\begin{array}{ccccc}
O & O & O & O & O \\
O & d_1 & P_{|+\rangle}(a_1) & P_{|0\rangle}(a_3) & d_3 \\
O & O & P_{|+\rangle}(a_5) & P_{|+\rangle}(a_7) & O \\
O & O & P_{|0\rangle}(a_6) & P_{|0\rangle}(a_8) & O \\
O & d_2 & P_{|+\rangle}(a_2) & P_{|0\rangle}(a_4) & d_4
\end{array}.$$

We found the operation and required time of ED_0 are the same as those of ED_+ and we omit the subscripts 0 or +.

Remark: after a logical Hadamard gate, the labels of data qubits 2 and 3 are switched. This can be fixed by applying appropriate SWAPs and it takes 2 more time steps in structures I or II. However, we don't adjust it until a CNOT gate is operating on two tiles with different labels.

CNOT gate Consider the $vhCNOT$ from $|q_1q_2q_3q_4\rangle$ to $|d_1d_2d_3d_4\rangle$.

Time step 0:

$$
\begin{array}{|ccccc|}
\hline
O & O & O & O & O \\
O & q_1 & O & O & q_3 \\
O & O & O & O & O \\
O & O & O & O & O \\
O & q_2 & O & O & q_4 \\
\hline
O & O & O & O & O \\
O & d_1 & O & O & d_3 \\
O & O & O & O & O \\
O & O & O & O & O \\
O & d_2 & O & O & d_4 \\
\hline
\end{array}
$$

Time step 1:

$$
\begin{array}{|ccccccc|cc}
\hline
O & & O & O & O & O & & & \\
O & \Leftrightarrow & q_1 & O & O & q_3 & & \Leftrightarrow & O \\
O & & O & O & O & O & & & \\
O & & O & O & O & O & & & \\
O & \Leftrightarrow & q_2 & O & O & q_4 & & \Leftrightarrow & O \\
\hline
O & & O & O & O & O & & & \\
 & & \Updownarrow & & & \Updownarrow & & & \\
O & & d_1 & O & O & d_3 & & & \\
O & & O & O & O & O & & & \\
O & & O & O & O & O & & & \\
 & & \Updownarrow & & & \Updownarrow & & & \\
O & & d_2 & O & O & d_4 & & & \\
\hline
\end{array}
$$

Time step 2:

$$
\begin{array}{|ccccc|c}
\hline
O & O & O & O & O & \\
q_1 & O & O & O & O & q_3 \\
\Updownarrow & & & & & \Updownarrow \\
O & O & O & O & O & O \\
O & O & O & O & O & \\
q_2 & O & O & O & O & q_4 \\
\hline
\Updownarrow & \Updownarrow & & & \Updownarrow & \Updownarrow \\
O & d_1 & O & O & d_3 & O \\
O & O & O & O & O & \\
O & O & O & O & O & \\
 & \Updownarrow & & & \Updownarrow & \\
O & d_2 & O & O & d_4 & \\
O & O & O & O & O & \\
\hline
\end{array}
$$

Time step 3:

$$
\begin{array}{|ccccc|c}
\hline
O & O & O & O & O & \\
O & O & O & O & O & \\
q_1 & O & O & O & O & q_3 \\
\Updownarrow & & & & & \Updownarrow \\
O & O & O & O & O & O \\
 & \Updownarrow & & & \Updownarrow & \\
O & d_1 & O & O & d_3 & \\
\hline
q_2 & O & O & O & O & q_4 \\
\Updownarrow & & & & & \Updownarrow \\
O & O & O & O & O & O \\
 & \Updownarrow & & & \Updownarrow & \\
O & d_2 & O & O & d_4 & \\
O & O & O & O & O & \\
O & O & O & O & O & \\
\hline
\end{array}
$$

Time step 4:

$$
\begin{array}{|cccccc|cc}
\hline
O & & O & O & O & O & & \\
O & & O & O & O & O & & \\
O & & O & O & O & O & & \\
q_1 & \rightarrow & d_1 & O & O & d_3 & \leftarrow & q_3 \\
O & & O & O & O & O & & \\
\hline
O & & O & O & O & O & & \\
q_2 & \rightarrow & d_2 & O & O & d_4 & \leftarrow & q_4 \\
O & & O & O & O & O & & \\
O & & O & O & O & O & & \\
O & & O & O & O & O & & \\
\hline
\end{array}
$$

Time step 5:

$$
\begin{array}{|ccccc|c}
\hline
O & O & O & O & O & \\
O & O & O & O & O & \\
O & O & O & O & O & O \\
\Updownarrow & & & & & \Updownarrow \\
q_1 & d_1 & O & O & d_3 & q_3 \\
 & \Updownarrow & & & \Updownarrow & \\
O & O & O & O & O & \\
\hline
O & O & O & O & O & O \\
\Updownarrow & & & & & \Updownarrow \\
q_2 & d_2 & O & O & d_4 & q_4 \\
 & \Updownarrow & & & \Updownarrow & \\
O & O & O & O & O & \\
O & O & O & O & O & \\
O & O & O & O & O & \\
\hline
\end{array}
$$

Time step 6:

$$
\begin{array}{|ccccc|c}
\hline
O & O & O & O & O & \\
O & O & O & O & O & O \\
\Updownarrow & & & & & \Updownarrow \\
q_1 & O & O & O & O & q_3 \\
O & O & O & O & O & \\
O & d_1 & O & O & d_3 & O \\
\hline
\Updownarrow & \Updownarrow & & & \Updownarrow & \Updownarrow \\
q_2 & O & O & O & O & q_4 \\
O & O & O & O & O & \\
O & d_2 & O & O & d_4 & \\
 & \Updownarrow & & & \Updownarrow & \\
O & O & O & O & O & \\
O & O & O & O & O & \\
\hline
\end{array}
$$

Time step 7:

$$
\begin{array}{|cccccc|cc}
\hline
O & & O & O & O & O & & \\
q_1 & \Leftrightarrow & O & O & O & O & \Leftrightarrow & q_3 \\
O & & O & O & O & O & & \\
O & & O & O & O & O & & \\
q_2 & \Leftrightarrow & O & O & O & O & \Leftrightarrow & q_4 \\
\hline
O & & d_1 & O & O & d_3 & & \\
 & & \Updownarrow & & & \Updownarrow & & \\
O & & O & O & O & O & & \\
O & & O & O & O & O & & \\
O & & d_2 & O & O & d_4 & & \\
 & & \Updownarrow & & & \Updownarrow & & \\
O & & O & O & O & O & & \\
\hline
\end{array}
$$

SWAP gate Consider the $vSWAP$: Time step 0:

$$
\begin{array}{|ccccc|}
\hline
O & O & O & O & O \\
O & O & O & O & O \\
O & O & O & O & O \\
O & O & O & O & O \\
O & O & O & O & O \\
\hline
O & O & O & O & O \\
O & d_1 & O & O & d_3 \\
O & O & O & O & O \\
O & O & O & O & O \\
O & d_2 & O & O & d_4 \\
\hline
\end{array}
$$

Time step 1:

$$
\begin{array}{|ccccc|}
\hline
O & O & O & O & O \\
O & O & O & O & O \\
O & O & O & O & O \\
O & O & O & O & O \\
O & O & O & O & O \\
\hline
O & O & O & O & O \\
 & \Updownarrow & & & \Updownarrow \\
O & d_1 & O & O & d_3 \\
O & O & O & O & O \\
O & O & O & O & O \\
 & \Updownarrow & & & \Updownarrow \\
O & d_2 & O & O & d_4 \\
\hline
\end{array}
$$

Time step 2:

$$
\begin{array}{|ccccc|}
\hline
O & O & O & O & O \\
O & O & O & O & O \\
O & O & O & O & O \\
O & O & O & O & O \\
O & O & O & O & O \\
\hline
 & \Updownarrow & & & \Updownarrow \\
O & d_1 & O & O & d_3 \\
O & O & O & O & O \\
O & O & O & O & O \\
 & \Updownarrow & & & \Updownarrow \\
O & d_2 & O & O & d_4 \\
O & O & O & O & O \\
\hline
\end{array}
$$

Time step 3:

$$
\begin{array}{|ccccc|}
\hline
O & O & O & O & O \\
O & O & O & O & O \\
O & O & O & O & O \\
O & O & O & O & O \\
 & \Updownarrow & & & \Updownarrow \\
O & d_1 & O & O & d_3 \\
\hline
O & O & O & O & O \\
O & O & O & O & O \\
 & \Updownarrow & & & \Updownarrow \\
O & d_2 & O & O & d_4 \\
O & O & O & O & O \\
O & O & O & O & O \\
\hline
\end{array}
$$

Time step 4:

$$
\begin{array}{|ccccc|}
\hline
O & O & O & O & O \\
O & O & O & O & O \\
O & O & O & O & O \\
 & \Updownarrow & & & \Updownarrow \\
O & d_1 & O & O & d_3 \\
O & O & O & O & O \\
\hline
O & O & O & O & O \\
 & \Updownarrow & & & \Updownarrow \\
O & d_2 & O & O & d_4 \\
O & O & O & O & O \\
O & O & O & O & O \\
O & O & O & O & O \\
\hline
\end{array}
$$

Time step 5:

$$
\begin{array}{|ccccc|}
\hline
O & O & O & O & O \\
O & O & O & O & O \\
 & \Updownarrow & & & \Updownarrow \\
O & d_1 & O & O & d_3 \\
O & O & O & O & O \\
O & O & O & O & O \\
\hline
 & \Updownarrow & & & \Updownarrow \\
O & d_2 & O & O & d_4 \\
O & O & O & O & O \\
O & O & O & O & O \\
O & O & O & O & O \\
O & O & O & O & O \\
\hline
\end{array}
$$

$P_{|0\rangle}$ **and** $P_{|+\rangle}$ The logical state preparation circuit $P_{|0\rangle}$ in Fig. 22: Time step 1:

$$\begin{array}{ccccc}
O & O & O & O & O \\
O & O & P_{|+\rangle}(a_1) & P_{|0\rangle}(a_3) & O \\
O & O & O & O & O \\
O & O & O & O & O \\
O & O & P_{|+\rangle}(a_2) & P_{|0\rangle}(a_4) & O
\end{array}$$

Time step 2:

$$\begin{array}{cccccc}
O & O & O & & O & O \\
O & O & a_1 & \rightarrow & a_3 & O \\
O & O & O & & O & O \\
O & O & O & & O & O \\
O & O & a_2 & \rightarrow & a_4 & O
\end{array}$$

Time step 3:

$$\begin{array}{ccccccccc}
O & O & & O & & O & & O \\
O & O & \Leftrightarrow & a_1 & & a_3 & \Leftrightarrow & O \\
O & O & & O & & O & & O \\
O & O & & O & & O & & O \\
O & O & \Leftrightarrow & a_2 & O & a_4 & \Leftrightarrow & O
\end{array}$$

The logical state preparation circuit $P_{|+\rangle}$ in Fig. 22: Time step 1:

$$\begin{array}{ccccc}
O & O & O & O & O \\
O & O & O & O & O \\
O & P_{|+\rangle}(a_1) & O & O & P_{|0\rangle}(a_3) \\
O & P_{|+\rangle}(a_2) & O & O & P_{|0\rangle}(a_4) \\
O & O & O & O & O
\end{array}$$

Time step 2:

$$
\begin{array}{ccccc}
O & O & O & O & O \\
O & O & O & O & O \\
O & a_1 & O & O & a_3 \\
 & \downarrow & & & \downarrow \\
O & a_2 & O & O & a_4 \\
O & O & O & O & O
\end{array}
$$

Time step 3:

$$
\begin{array}{ccccc}
O & O & O & O & O \\
O & O & O & O & O \\
 & \Updownarrow & & & \Updownarrow \\
O & a_1 & O & O & a_3 \\
O & a_2 & O & O & a_4 \\
 & \Updownarrow & & & \Updownarrow \\
O & O & O & O & O
\end{array}
$$

decoding circuit

The decoding circuit of the C_4 code when the spectator state is $|0\rangle$ in Fig. 26: Time step 1:

$$
\begin{array}{ccccccc}
O & O & & O & O & & O \\
O & q_1 & \Leftrightarrow & O & O & \Leftrightarrow & q_3 \\
O & O & & O & O & & O \\
O & O & & O & O & & O \\
O & q_2 & \Leftrightarrow & O & O & \Leftrightarrow & q_4
\end{array}
$$

Time step 2:

$$
\begin{array}{cccccc}
O & O & O & & O & O \\
O & O & q_1 & \leftarrow & q_3 & O \\
O & O & O & & O & O \\
O & O & O & & O & O \\
O & O & q_2 & \leftarrow & q_4 & O
\end{array}
$$

Time step 3:

$$\begin{array}{ccccc}
O & O & O & O & O \\
O & O & q_1 & q_3 & O \\
 & & \Updownarrow & & \\
O & O & O & O & O \\
O & O & O & O & O \\
 & & \Updownarrow & & \\
O & O & q_2 & q_4 & O
\end{array}$$

Time step 4:

$$\begin{array}{ccccc}
O & O & O & O & O \\
O & O & O & q_3 & O \\
O & O & q_1 & O & O \\
 & & \downarrow & & \\
O & O & q_2 & O & O \\
O & O & O & q_4 & O
\end{array}$$

The decoding circuit when the spectator state is $|+\rangle$ in Fig. 26: Time step 1:

$$\begin{array}{ccccc}
O & O & O & O & O \\
O & q_1 & O & O & q_3 \\
 & \Updownarrow & & & \Updownarrow \\
O & O & O & O & O \\
O & O & O & O & O \\
 & \Updownarrow & & & \Updownarrow \\
O & q_2 & O & O & q_4
\end{array}$$

Time step 2:

$$\begin{array}{ccccc}
O & O & O & O & O \\
O & O & O & O & O \\
O & q_1 & O & O & q_3 \\
 & \downarrow & & & \downarrow \\
O & q_2 & O & O & q_4 \\
O & O & O & O & O
\end{array}$$

Time step 3:

$$\begin{array}{ccccccc}
O & O & & O & O & & O \\
O & O & & O & O & & O \\
O & q_1 & \Leftrightarrow & O & O & \Leftrightarrow & q_3 \\
O & q_2 & & O & O & & q_4 \\
O & O & & O & O & & O
\end{array}$$

Time step 4:

$$\begin{array}{cccccc}
O & O & O & & O & O \\
O & O & O & & O & O \\
O & O & q_1 & \leftarrow & q_3 & O \\
O & q_2 & O & & O & q_4 \\
O & O & O & & O & O
\end{array}$$

S **and** T **gates** Let $|q_1q_2q_3q_4\rangle$ denote the $|\overline{+i}\rangle$ state and $|d_1d_2d_3d_4\rangle$ be the data qubit in Fig. 13.
Implementation of the S gate:
Time step 1:

$$\begin{array}{|cccccc|cc}
\hline
O & & O & O & O & O & & \\
O & \Leftrightarrow & q_1 & O & O & q_3 & \Leftrightarrow & O \\
O & & O & O & O & O & & \\
O & & O & O & O & O & & \\
O & \Leftrightarrow & q_2 & O & O & q_4 & \Leftrightarrow & O \\
\hline
O & & O & O & O & O & & \\
O & & d_1 & O & O & d_3 & & \\
O & & O & O & O & O & & \\
O & & O & O & O & O & & \\
O & & d_2 & O & O & d_4 & & \\
\hline
\end{array}$$

Time step 2:

O	O	O	O	O	
q_1	O	O	O	O	q_3
$\Updownarrow$					$\Updownarrow$
O	O	O	O	O	O
O	O	O	O	O	
q_2	O	O	O	O	q_4
$\Updownarrow$					$\Updownarrow$
O	O	O	O	O	O
O	d_1	O	O	d_3	
O	O	O	O	O	
O	O	O	O	O	
O	d_2	O	O	d_4	

Time step 3:

O	O	O	O	O	
O	O	O	O	O	
q_1	O	O	O	O	q_3
$\Updownarrow$					$\Updownarrow$
O	O	O	O	O	O
O	O	O	O	O	
q_2	O	O	O	O	q_4
$\Updownarrow$					$\Updownarrow$
O	d_1	O	O	d_3	O
O	O	O	O	O	
O	O	O	O	O	
O	d_2	O	O	d_4	

Time step 4:

$$
\begin{array}{|ccccc|c}
\hline
O & O & O & O & O & \\
O & O & O & O & O & \\
O & O & O & O & O & \\
q_1 & O & O & O & O & q_3 \\
\Updownarrow & & & & & \Updownarrow \\
O & O & O & O & O & O \\
\hline
O & O & O & O & O & \\
q_2 & d_1 & O & O & d_3 & q_4 \\
\Updownarrow & & & & & \Updownarrow \\
O & O & O & O & O & O \\
O & O & O & O & O & \\
O & d_2 & O & O & d_4 & \\
\hline
\end{array}
$$

Time step 5:

$$
\begin{array}{|ccccc|c}
\hline
O & O & O & O & O & \\
O & O & O & O & O & \\
O & O & O & O & O & \\
O & O & O & O & O & \\
q_1 & O & O & O & O & q_3 \\
\Updownarrow & & & & & \Updownarrow \\
\hline
O & O & O & O & O & O \\
O & d_1 & O & O & d_3 & \\
q_2 & O & O & O & O & q_4 \\
\Updownarrow & & & & & \Updownarrow \\
O & O & O & O & O & O \\
O & d_2 & O & O & d_4 & \\
\hline
\end{array}
$$

Time step 6:

O	O	O	O	O	
O	O	O	O	O	
O	O	O	O	O	
O	O	O	O	O	
O	O	O	O	O	
q_1	O	O	O	O	q_3
$\Updownarrow$					$\Updownarrow$
O	d_1	O	O	d_3	O
O	O	O	O	O	
q_2	O	O	O	O	q_4
$\Updownarrow$					$\Updownarrow$
O	d_2	O	O	d_4	O

Time step 7:

O		O	O	O	O		
O		O	O	O	O		
O		O	O	O	O		
O		O	O	O	O		
O		O	O	O	O		
O		O	O	O	O		
q_1	$\leftarrow$	d_1	O	O	d_3	$\rightarrow$	q_3
O		O	O	O	O		
O		O	O	O	O		
q_2	$\leftarrow$	d_2	O	O	d_4	$\rightarrow$	q_4

Time step 8:

$$
\begin{array}{|ccccc|c}
\hline
O & O & O & O & O & \\
O & O & O & O & O & \\
O & O & O & O & O & \\
O & O & O & O & O & \\
O & O & O & O & O & \\
\hline
O & O & O & O & O & \\
H(q_1) & d_1 & O & O & d_3 & H(q_3) \\
O & O & O & O & O & \\
O & O & O & O & O & \\
H(q_2) & d_2 & O & O & d_4 & H(q_4) \\
\hline
\end{array}
$$

Time step 9:

$$
\begin{array}{|cccccc|cc}
\hline
O & & O & O & O & O & & \\
O & & O & O & O & O & & \\
O & & O & O & O & O & & \\
O & & O & O & O & O & & \\
O & & O & O & O & O & & \\
\hline
O & & O & O & O & O & & \\
q_1 & \leftarrow & d_1 & O & O & d_3 & \rightarrow & q_3 \\
O & & O & O & O & O & & \\
O & & O & O & O & O & & \\
q_2 & \leftarrow & d_2 & O & O & d_4 & \rightarrow & q_4 \\
\hline
\end{array}
$$

Time step 10:

O	O	O	O	O	
O	O	O	O	O	
O	O	O	O	O	
O	O	O	O	O	
O	O	O	O	O	
O	O	O	O	O	
$H(q_1)$	d_1	O	O	d_3	$H(q_3)$
O	O	O	O	O	
O	O	O	O	O	
$H(q_2)$	d_2	O	O	d_4	$H(q_4)$

Now let $|q_1q_2q_3q_4\rangle$ denote the $T\,|\mp\rangle$ state and $|d_1d_2d_3d_4\rangle$ be the data qubit in Fig. 18.
Implementation of the T gate:
Time step 1:

O		O	O	O	O		
O	$\Leftrightarrow$	q_1	O	O	q_3	$\Leftrightarrow$	O
O		O	O	O	O		
O		O	O	O	O		
O	$\Leftrightarrow$	q_2	O	O	q_4	$\Leftrightarrow$	O
O		O	O	O	O		
O		d_1	O	O	d_3		
O		O	O	O	O		
O		O	O	O	O		
O		d_2	O	O	d_4		

Time step 2:

$$
\begin{array}{|ccccc|c}
\hline
O & O & O & O & O & \\
q_1 & O & O & O & O & q_3 \\
\Updownarrow & & & & & \Updownarrow \\
O & O & O & O & O & O \\
O & O & O & O & O & \\
q_2 & O & O & O & O & q_4 \\
\hline
\Updownarrow & & & & & \Updownarrow \\
O & O & O & O & O & O \\
O & d_1 & O & O & d_3 & \\
O & O & O & O & O & \\
O & O & O & O & O & \\
O & d_2 & O & O & d_4 & \\
\hline
\end{array}
$$

Time step 3:

$$
\begin{array}{|ccccc|c}
\hline
O & O & O & O & O & \\
O & O & O & O & O & \\
q_1 & O & O & O & O & q_3 \\
\Updownarrow & & & & & \Updownarrow \\
O & O & O & O & O & O \\
\hline
O & O & O & O & O & \\
q_2 & O & O & O & O & q_4 \\
\Updownarrow & & & & & \Updownarrow \\
O & d_1 & O & O & d_3 & O \\
O & O & O & O & O & \\
O & O & O & O & O & \\
O & d_2 & O & O & d_4 & \\
\hline
\end{array}
$$

Time step 4:

$$\begin{array}{|ccccc|c}
\hline
O & O & O & O & O & \\
O & O & O & O & O & \\
O & O & O & O & O & \\
q_1 & O & O & O & O & q_3 \\
\Updownarrow & & & & & \Updownarrow \\
O & O & O & O & O & O \\
\hline
O & O & O & O & O & \\
q_2 & d_1 & O & O & d_3 & q_4 \\
\Updownarrow & & & & & \Updownarrow \\
O & O & O & O & O & O \\
O & O & O & O & O & \\
O & d_2 & O & O & d_4 & \\
\hline
\end{array}$$

Time step 5:

$$\begin{array}{|ccccc|c}
\hline
O & O & O & O & O & \\
O & O & O & O & O & \\
O & O & O & O & O & \\
O & O & O & O & O & \\
q_1 & O & O & O & O & q_3 \\
\Updownarrow & & & & & \Updownarrow \\
\hline
O & O & O & O & O & O \\
O & d_1 & O & O & d_3 & \\
q_2 & O & O & O & O & q_4 \\
\Updownarrow & & & & & \Updownarrow \\
O & O & O & O & O & O \\
O & d_2 & O & O & d_4 & \\
\hline
\end{array}$$

Time step 6:

$$
\begin{array}{|ccccc|c}
\hline
O & O & O & O & O & \\
O & O & O & O & O & \\
O & O & O & O & O & \\
O & O & O & O & O & \\
O & O & O & O & O & \\
\hline
q_1 & O & O & O & O & q_3 \\
\Updownarrow & & & & & \Updownarrow \\
O & d_1 & O & O & d_3 & O \\
O & O & O & O & O & \\
q_2 & O & O & O & O & q_4 \\
\Updownarrow & & & & & \Updownarrow \\
O & d_2 & O & O & d_4 & O \\
\hline
\end{array}
$$

Time step 7:

$$
\begin{array}{|cccccc|cc}
\hline
O & & O & O & O & O & & \\
O & & O & O & O & O & & \\
O & & O & O & O & O & & \\
O & & O & O & O & O & & \\
O & & O & O & O & O & & \\
\hline
O & & O & O & O & O & & \\
q_1 & \rightarrow & d_1 & O & O & d_3 & \leftarrow & q_3 \\
O & & O & O & O & O & & \\
O & & O & O & O & O & & \\
q_2 & \rightarrow & d_2 & O & O & d_4 & \leftarrow & q_4 \\
\hline
\end{array}
$$

Time step 8:

$$
\begin{array}{|ccccc|c}
\hline
O & O & O & O & O & \\
O & O & O & O & O & \\
O & O & O & O & O & \\
O & O & O & O & O & \\
O & O & O & O & O & \\
\hline
O & O & O & O & O & \\
q_1 & M_Z(d_1) & O & O & M_Z(d_3) & q_3 \\
O & O & O & O & O & \\
O & O & O & O & O & \\
q_2 & M_Z(d_2) & O & O & M_Z(d_4) & q_4 \\
\hline
\end{array}
$$

Time step 9:

$$
\begin{array}{|cccccc|cc}
\hline
O & & O & O & O & O & & \\
O & & O & O & O & O & & \\
O & & O & O & O & O & & \\
O & & O & O & O & O & & \\
O & & O & O & O & O & & \\
\hline
O & & O & O & O & O & & \\
q_1 & \Leftrightarrow & d_1 & O & O & d_3 & \Leftrightarrow & q_3 \\
O & & O & O & O & O & & \\
O & & O & O & O & O & & \\
q_2 & \Leftrightarrow & d_2 & O & O & d_4 & \Leftrightarrow & q_4 \\
\hline
\end{array}
$$

Then an S gate is applied.

18 Preparation of a logical state of the Knill's post-selection scheme at the top level

Recall that the tile of the Steane code is a 6×8 square lattice as follows:

O	O	O	O	O	O	O	O
O	d_6	O	d_5	O	d_3	O	O
O	O	O	O	O	O	O	O
O	O	O	O	O	O	O	O
O	O	O	O	O	O	O	O
O	d_4	O	d_2	O	d_1	O	d_7

(Note that $|q_1\rangle$, $|q_2\rangle$, $|q_3\rangle$, $|q_4\rangle$, $|q_5\rangle$, $|q_6\rangle$, and $|q_7\rangle$ in Fig. 24 are reordered as $|d_6\rangle$, $|d_7\rangle$, $|d_1\rangle$, $|d_5\rangle$, $|d_4\rangle$, $|d_2\rangle$, and $|d_3\rangle$) Preparation of the logical state $|0\rangle$ of the concatenation of C_{ed}^m and C_{ec}:

Time step 1:

O	O	O	O	O	O	O	O
O	O	O	O	O	O	O	O
O	O	O	O	$P_{\|+\rangle}(d_6)$	$P_{\|0\rangle}(d_1)$	O	O
O	O	O	$P_{\|+\rangle}(d_5)$	$P_{\|0\rangle}(d_3)$	O	O	O
O	O	O	$P_{\|0\rangle}(d_2)$	$P_{\|+\rangle}(d_7)$	O	O	O
O	O	O	O	O	O	O	O

Time step 2:

O	O	O	O	O	O	O	O
O	O	O	O	O	O	O	O
O	O	O	$P_{\|0\rangle}(d_4)$	$d_6 \rightarrow$	d_1	O	O
O	O	O	d_5 $\downarrow$	d_3	O	O	O
O	O	O	d_2	$\uparrow$ d_7	O	O	O
O	O	O	O	O	O	O	O

Time step 3:

$$
\begin{array}{ccccccccc}
\hline
O & O & O & O & & O & O & O & O \\
O & O & O & O & & O & O & O & O \\
O & O & O & d_4 & \leftarrow & d_6 & d_1 & O & O \\
O & O & O & d_5 & \rightarrow & d_3 & O & O & O \\
O & O & O & d_2 & \leftarrow & d_7 & O & O & O \\
O & O & O & O & & O & O & O & O \\
\hline
\end{array}
$$

Time step 4:

$$
\begin{array}{ccccccccc}
\hline
O & O & O & O & O & & O & O & O \\
O & O & O & O & O & & O & O & O \\
O & O & O & d_4 & d_6 & & d_1 & O & O \\
 & & & \uparrow & \downarrow & & \Updownarrow & & \\
O & O & O & d_5 & d_3 & & O & O & O \\
O & O & O & d_2 & d_7 & \Leftrightarrow & O & O & O \\
O & O & O & O & O & & O & O & O \\
\hline
\end{array}
$$

Time step 5:

$$
\begin{array}{ccccccccc}
\hline
O & O & O & & O & O & O & O & O \\
O & O & O & & O & O & O & O & O \\
 & & & & & \Updownarrow & & & \\
O & O & O & \Leftrightarrow & d_4 & d_6 & O & O & O \\
O & O & O & & d_5 & d_3 & d_1 & O & O \\
 & & & & & & \uparrow & & \\
O & O & O & & d_2 & O & d_7 & O & O \\
 & & & & \Updownarrow & & & & \\
O & O & O & & O & O & O & O & O \\
\hline
\end{array}
$$

Time step 6:

$$
\begin{array}{ccccccccccc}
\hline
O & O & & O & O & & O & O & & O & O \\
O & O & & O & O & \Leftrightarrow & d_6 & O & & O & O \\
O & O & \Leftrightarrow & d_4 & O & & O & O & & O & O \\
 & & & & \Updownarrow & & \Updownarrow & & & & \\
O & O & & O & d_5 & & d_3 & d_1 & & O & O \\
O & O & & O & O & & O & d_7 & \Leftrightarrow & O & O \\
O & O & & O & d_2 & & O & O & & O & O \\
\hline
\end{array}
$$

Time step 7:

$$
\begin{array}{cccccccccc}
\hline
O & O & O & & O & O & O & O & & O \\
O & O & O & \Leftrightarrow & d_6 & O & O & O & & O \\
 & & & & & \Updownarrow & & & & \\
O & d_4 & O & & d_5 & d_3 & O & O & & O \\
 & \Updownarrow & & & & & & & & \\
O & O & O & & O & O & d_1 & O & & O \\
 & & & & & & \Updownarrow & & & \\
O & O & O & & O & O & O & d_7 & \Leftrightarrow & O \\
O & O & O & & d_2 & O & O & O & & O \\
\hline
\end{array}
$$

Time step 8:

$$
\begin{array}{cccccccccc}
\hline
O & O & & O & O & O & & O & O & O \\
O & O & \Leftrightarrow & d_6 & O & d_3 & \Leftrightarrow & O & O & O \\
 & & & & \Updownarrow & & & & & \\
O & O & & O & d_5 & O & & O & O & O \\
O & d_4 & & O & O & O & & O & O & O \\
 & \Updownarrow & & & & & & & & \\
O & O & & O & O & O & & d_1 & O & d_7 \\
 & & & & & & & \Updownarrow & & \Updownarrow \\
O & O & & O & d_2 & O & & O & O & O \\
\hline
\end{array}
$$

Time step 9:

O	O	O	O	O	O	O	O
O	d_6	O	d_5	O	d_3	O	O
O	O	O	O	O	O	O	O
O	O	O	O	O	O	O	O
O	d_4	O	O	O	O	O	O
	$\Updownarrow$						
O	O	O	d_2	O	d_1	O	d_7